Titre: Title:	Une approche multi-physiques et multi-échelles pour l'amélioration de l'efficacité de la modélisation et de la simulation des disjoncteurs haute-tension
Auteur: Author:	Guillaume Pernaudat
Date:	2017
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Pernaudat, G. (2017). Une approche multi-physiques et multi-échelles pour l'amélioration de l'efficacité de la modélisation et de la simulation des disjoncteurs haute-tension [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/2849/

UNIVERSITÉ DE MONTRÉAL

UNE APPROCHE MULTI-PHYSIQUES ET MULTI-ÉCHELLES POUR L'AMÉLIORATION DE L'EFFICACITÉ DE LA MODÉLISATION ET DE LA SIMULATION DES DISJONCTEURS HAUTE-TENSION

GUILLAUME PERNAUDAT
DÉPARTEMENT DE GÉNIE MÉCANIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(GÉNIE MÉCANIQUE)
DÉCEMBRE 2017

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

UNE APPROCHE MULTI-PHYSIQUES ET MULTI-ÉCHELLES POUR L'AMÉLIORATION DE L'EFFICACITÉ DE LA MODÉLISATION ET DE LA SIMULATION DES DISJONCTEURS HAUTE-TENSION

présentée par : PERNAUDAT Guillaume
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

M. REGGIO Marcelo, Ph. D., président
M. TRÉPANIER Jean-Yves, Ph. D., membre et directeur de recherche
M. CAMARERO Ricardo, Ph. D., membre et codirecteur de recherche
M. ROBIN-JOUAN Philippe, Ph. D., membre et codirecteur de recherche
M. GUIBAULT François, Ph. D., membre
M. PARASCHIVOIU Marius, Ph. D., membre externe

DÉDICACE

À mes parents, mon frère et ma soeur...

RÉSUMÉ

La conception des disjoncteurs haute-tension est une tâche complexe et difficile en raison des écoulements internes compressibles supersoniques à haute température et instationnaires, des forts champs électriques, magnétiques et radiatifs. L'arc électrique créé lors de la séparation des contacts électriques en est la cause principale. Il s'apparente à un plasma porté à des températures qui avoisinent les 30 000 K et qui provoque la sublimation des matériaux à sa proximité. Devant la complexité de ces phénomènes et pour pouvoir en tirer des critères de design, les outils de simulations numériques sont indispensables pour répondre aux besoins des ingénieurs. Cependant, la modélisation et la simulation numérique de ces phénomènes multi-physiques, fortement non-linéaires, comportent de nombreux défis de natures multidisciplinaires. En effet, les aspects concernant : la physique, les méthodes numériques, la modélisation géométrique et le calcul haute-performance sont fortement entremêlés. La contrainte du temps de calcul, qui peut s'échelonner sur plusieurs semaines, est un frein supplémentaire majeur lors du développement de l'appareillage de coupure des circuits électriques haute-tension. Par conséquent, l'efficacité de la modélisation et de la simulation des disjoncteurs haute-tension dépend d'une approche multi-physiques et multi-échelles dans la conception des outils scientifiques. Dans un partenariat entre l'École Polytechnique de Montréal et la société Alstom (désormais intégrée à General Electric), vieux de plus de 30 ans, l'outil scientifique MC^3 a été développé et enrichi par de nouveaux modèles au fil du temps afin de répondre aux besoins de simulation. En revanche, il était nécessaire de le moderniser : son architecture n'était pas adaptée au contexte multi-physiques, les algorithmes n'étaient pas adaptés aux architectures des matériels informatiques modernes, les performances en temps de calcul n'étaient pas maximisées et les simulations étaient restreintes à des calculs 2D axisymétriques. Dans un but de modernisation de MC^3 et d'amélioration de son efficacité du point de vue des performances et de la fidélité des résultats de calcul, quatre axes de recherche ont été poursuivis consécutivement.

En premier lieu, une révision des modèles a été proposée pour s'assurer de leur pertinence. Une mise à jour mineure du schéma numérique utilisé pour la résolution de la mécanique des fluides a été effectuée. Les autres modèles étaient à jour vis-à-vis de l'état de l'art. Cela a constitué des fondations stables et solides pour supporter la suite des étapes de modernisation.

En second lieu, les performances en vitesse d'exécution avaient besoin d'être améliorées, une analyse approfondie de l'outil de calcul existant a donc été réalisée. Le profilage du code et l'analyse numérique ont permis d'orienter les choix de modifications qui ont été réalisé

pour l'accélération de la simulation des physiques. Cette accélération a pu être obtenue en modifiant considérablement les algorithmes, la structure des modules physiques, en vectorisant et en parallélisant les boucles de calcul les plus consommatrices de temps. Devant la quantité de changements à apporter, il a été choisi de réécrire entièrement l'application en C++ et en assembleur pour les portions les plus critiques en termes de performances. Suivant les physiques, les accélérations mesurées ont pu atteindre un facteur compris entre trente et six-cents fois.

En troisième lieu, une nouvelle architecture logicielle a été créée pour exposer au maximum le parallélisme de tâches et apporter une interconnexion efficace entre les modules qui est adaptée au contexte multiphysiques. Les résultats des premiers tests de performance nous ont permis d'estimer le facteur d'accélération globale par rapport à MC^3 à une valeur comprise entre trente et cent fois. Cette contribution fait avancer le domaine de la simulation numérique des disjoncteurs haute-tension en proposant une architecture simple à maintenir et accessible et qui est efficace en terme de performance.

En dernier lieu, on a souhaité améliorer la prédiction des calculs vis-à-vis des écoulements de gaz dans les chambres de disjoncteurs à haute-tension. Pour cela, une extension aux calculs 3D de mécanique des fluides a été ajoutée. Cependant, afin de maximiser l'efficacité, on souhaitait conserver un maximum de volumes résolus selon des hypothèses 2D axisymétriques. Une méthode de couplage adaptée au transfert bidirectionnel d'informations entre des régions 2D axisymétriques et 3D a donc été développée. Ainsi, des régions 3D sont insérées à des endroits judicieux et couplées avec d'autres régions 2D axisymétriques, ce qui représente un bon compromis entre la performance et la précision des calculs. Les validations numériques ont démontré de très bons résultats, l'écoulement est conservatif à travers l'interface de couplage et les résultats entre une solution hybride et une solution pleinement 3D sont difficilement différenciables. La méthode proposée se base sur une approche volumes finis et est indépendante du schéma numérique, ce qui la rend applicable à un large spectre d'applications en mécanique des fluides ou dans d'autres champs.

Finalement, le travail de recherche présenté dans cette thèse, a permis d'apporter une réduction importante du temps d'exécution et d'améliorer la prédiction des calculs pour la simulation numérique des disjoncteurs haute-tension. Parallèlement, la qualité logicielle de MC^3 a grandement progressé et les évolutions futures et la maintenabilité seront beaucoup plus aisées qu'avec l'ancien outil.

ABSTRACT

High-voltage circuit breaker design is a complex and challenging task, because of several factors including the dynamics of fluid flow (unsteady supersonic compressible internal fluid flow at high temperature) and the the strong coupled electric, magnetic and radiative fields. These phenomena are caused by the electric arc created by the separation of the electrodes. The arc is considered as a plasma at a temperature approaching 30,000 K and creating wall sublimation of the boundary materials. Facing the complexity of these phenomena, numerical simulation tools are indispensable to comply with engineering requirements and to obtain design criteria. However, modelling and numerical simulation of these multiphysics phenomena, which are highly non-linear, constitute multidisciplinary challenges. Indeed, physics, numerical analysis, geometric modelling and high performance computing are strongly interrelated. Calculation times can take several weeks, representing an additional obstacle to the development of high-voltage circuit breakers. Consequently, the efficiency of the modeling and the numerical simulation of high-voltage circuit breakers depends on a multiphysics and multiscale approach while designing scientific tools. As a result of a long term partnership, over 30 years, between Alstom company (now part of General Electric) and Ecole Polytechnique de Montréal, the scientific tool MC^3 has been developed and enriched with new models, following the simulation requirements. During the task of modernizing MC^3, it was found that the software architecture was not well adapted to the multiphysics context, that the algorithms were not suitable to modern computing hardware architecture, and that performance in terms of the calculation runtime were not maximized. In addition, simulations were limited to 2D axisymmetric cases. The research aims to address those issues to improve its efficiency from performance and result fidelity perspectives, and this has led to four research axes.

First, a model review has been conducted to confirm the relevance of each physical model. A minor update for the numerical scheme used to solve fluid dynamics has been performed. The other models were state of the art. That step constituted robust and stable foundations to support the consecutive steps of code modernization.

Secondly, performance in terms of the calculation runtime needed to be drastically improved. This has led to an in-depth profiling of the actual tool. Code profiling and numerical analysis have permited to steer modification choices which have been carried out to speedup the simulation for all the physics, independently. This speedup has been sucessfully achieved by significantly modifying the algorithms, physics module structures, vectorizing and parallelis-

ing the most time-consuming loops. In the face of the necessary code changes, a complete overhaul code was decided, using C++ language and assembler code for the most critical parts. Depending on the physics, the speedup reached a factor between thirty and six hundred with respect to the original code.

Thirdly, a new software architecture has been created to expose task parallelism and to bring multiphysics at the forefront and efficient interconnections between modules. Initial test results demonstrated good performance, the global estimated speedup factor, in comparison with MC^3, reached a factor between thirty and one hundred. This contribution permits to advance high-voltage circuit breaker numerical simulation domain by the proposal of a well understandable and simple to maintain architecture which is efficient in term of performance and easily reusable.

The last scientific research axis concerns fluid flow prediction improvement regarding to high-voltage circuit breaker chambers. For that purpose, an extension to 3D fluid flow calculation has been integrated. Nevertheless, while a full 3D simulation is still not feasible presently, the approach was to keep as much as possible the 2D axisymmetric volumes to maximize the overall efficiency. So, a coupling method adapted to bidirectional information transfers between 2D axisymmetric and 3D regions has been developed. Thus, 3D regions are inserted where wise and coupled with other 2D axisymmetric regions, thereby ensuring a good compromise between performance and precision of calculations. Numerical validations have shown very good results, fluid flow is conservative through the coupling interface and hybrid solution results versus fully 3D results are barely distinguishable. The proposed method is based on Finite Volumes, and is independent of the numerical scheme, which makes it applicable to a wide spectrum of applications, either in numerical fluid dynamics or in other fields.

Finally, the research presented in this thesis, has permited to decrease the calculation runtime and to improve result prediction for the numerical simulation of high-voltage circuit breakers. Alongside, the software quality of MC^3 increased very much and future evolutions and maintenability will be more comfortably done than ever.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

AGU	Address Generation Unit
ALE	Arbitrary Lagrandian Eulerian
AVX512	Advanced Vector eXtensions 512 bits
CFD	Computational Fluid Dynamics
ETL	Equilibre Thermodynamique Local
FVM	Finite Volume Method
GCL	Geometric Conservation Laws
HPC	High Performance Computing
HVCB	High-Voltage Circuit Breaker
ILP	Instruction Level Parallelism
NUMA	Non-Uniform Memory Access
RTE	Radiative Transport Equation
TLP	Task Level Parallelism
VLP	Vector Level Parallelism

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

Dans cette thèse, on propose des développements scientifiques en vue d'améliorer les modèles numériques qui permettent de simuler les écoulements gazeux dans les disjoncteurs haute-tension. Une approche de modélisation multi-physiques et multi-échelles orientera ces développements.

Contexte

Cette thèse s'inscrit dans le contexte d'une collaboration scientifique qui a débuté il y a près de 30 ans entre le groupe GRMIAO de l'École Polytechnique de Montréal et ALSTOM-Grid (nouvellement intégré à General Electric - Grid Solutions). Cette collaboration a permis le développement d'un logiciel de calcul scientifique, nommé MC^3, dédié à l'analyse et à la conception sur ordinateur des disjoncteurs haute-tension. MC^3 est un outil mature pour la simulation numérique des disjoncteurs mais il doit continuer d'évoluer pour être plus efficient, c'est-à-dire plus complet d'un point de vue de ses modèles physiques et aussi plus efficace en terme de temps de calcul et de précision. Le présent projet fait partie d'un programme de recherche sur plusieurs années visant à pérenniser cette collaboration.

Enjeux

En voulant modéliser la physique et simuler numériquement le fonctionnement d'un disjoncteur haute-tension, on se heurte à de nombreux défis. Ceux-ci sont principalement de trois natures : physique, numérique/algorithmique et informatique. En effet, on doit tout d'abord comprendre les phénomènes multi-physiques et fortement couplés qui entrent en jeu lors de l'opération d'un disjoncteur. Ensuite, la modélisation numérique de ces phénomènes par des techniques de simulation numérique modernes, précises et rapides en terme de temps d'exécution est le second défi auquel on doit répondre. Enfin, l'usage de techniques et technologies de programmation informatique avancées est essentiel pour obtenir une efficacité du calcul optimale sur les architectures modernes. Ces trois défis sont interreliés d'où la nécessité d'une approche globale à la conception et à la mise en œuvre d'un outil efficient.

Axes de recherche et organisation du document

Afin de répondre à ces trois défis, on doit tout d'abord s'assurer que les modèles physiques employés dans l'outil MC^3 soient encore pertinents vis-à-vis de l'état de l'art de la modélisation des disjoncteurs haute-tension. Une revue critique de littérature sera présentée dans le second chapitre en combinaison avec une description du fonctionnement des disjoncteurs haute-tension. Les objectifs spécifiques de la thèse seront énoncés à la fin du second chapitre. La sélection des modèles pertinents et les détails d'implémentation correspondants dans MC^3 seront détaillés dans le troisième chapitre.

Le premier axe de recherche, présenté au quatrième chapitre, concerne l'analyse approfondie de l'outil de calcul MC^3 afin d'appuyer les changements algorithmiques également décrits dans ce même chapitre, et les nouvelles implémentations des modèles sélectionnés. Cette étape franchie, les choix des modèles ainsi que leurs implémentations auront été optimisés tant sur les plans de la physique, des méthodes numériques, des techniques de calcul haute-performance que des paradigmes de programmation avancés.

La base structurelle globale de l'outil de calcul doit prendre en compte les couplages multi-physiques présents afin de minimiser les temps de calcul tout en facilitant la maintenance future du logiciel. Originellement, MC^3 n'avait pas été conçu dans cette optique. Dans un deuxième axe de recherche, une architecture globale de solveur en adéquation avec les couplages multi-physiques rencontrés dans les disjoncteurs et avec les défis associés sera proposée et constituera le cinquième chapitre de ce document. Cet axe de recherche est fortement lié au premier ; en effet, chaque physique ne peut pas être optimisée indépendamment des autres puisqu'elles doivent mettre en commun des calculs, des dépendances de données ou des séquences d'exécution bien précises.

Ainsi constitué d'une base structurelle adaptée, et d'algorithmes performants, le code de calcul est mieux prédisposé à l'ajout de nouveaux modèles. Dans le but d'améliorer la fidélité des calculs numériques en comparaison avec les résultats d'essais, un troisième axe de recherche est introduit dans le sixième chapitre. Cet axe de recherche intègre l'aspect multi-échelle des phénomènes physiques entrant en jeu lors de l'opération d'un disjoncteur haute-tension. Une méthodologie de couplage spatial multi-échelles 2D axisymétrique-3D pour les écoulements fluides y est proposée.

CHAPITRE 2 LES DISJONCTEURS HAUTE-TENSION : FONCTIONNEMENT, MODÉLISATION NUMÉRIQUE ET OBJECTIFS DE LA THÈSE

2.1 Les disjoncteurs haute-tension

Un disjoncteur haute-tension est un élément clef dans les réseaux de transport d'électricité. Ses rôles sont multiples : il doit assurer le passage du courant durant les phases normales d'opération ; il doit aussi l'interrompre en cas d'incident anormal sur le réseau ; enfin dans d'autres cas, il doit simplement établir la circulation du courant lors de la connexion d'une ligne électrique (jusqu'à 1200kV). Le rôle de protection du disjoncteur intervient en particulier lors de l'interruption d'un courant de court-circuit qui peut dépasser dans certains cas 300kA. Lors de l'ouverture du circuit électrique par le disjoncteur, un arc électrique se forme entre les contacts électriques qui s'éloignent. Pour que la coupure soit effective, l'arc doit être éteint. Le choix du milieu dans lequel se produit l'arc est d'une grande importance puisque de celui-ci dépend la réussite de la coupure. En effet, pour éteindre l'arc on lui imprime un écoulement suffisamment intense pour le refroidir et ainsi rendre le milieu de nouveau isolant électriquement. Le milieu choisi est fréquemment le gaz SF6 puisqu'il dispose de bonnes propriétés diélectriques (isolation), chimiques (stabilité à haute température) et thermiques (conductivité thermique - refroidissement). De plus, après la coupure, les molécules du SF6 dissociées par le plasma d'arc à forte température (environ 25000K) se recombinent, c'est à dire que le gaz SF6 se régénère de lui-même.

2.1.1 Configuration d'un disjoncteur

Afin de comprendre les difficultés de modélisation, nous allons tout d'abord aborder la configuration des disjoncteurs haute-tension. La figure 2.1 illustre la complexité géométrique de ces appareils avec de nombreux détails géométriques nécessaires à la fiabilité du fonctionnement et à la réussite de la coupure.

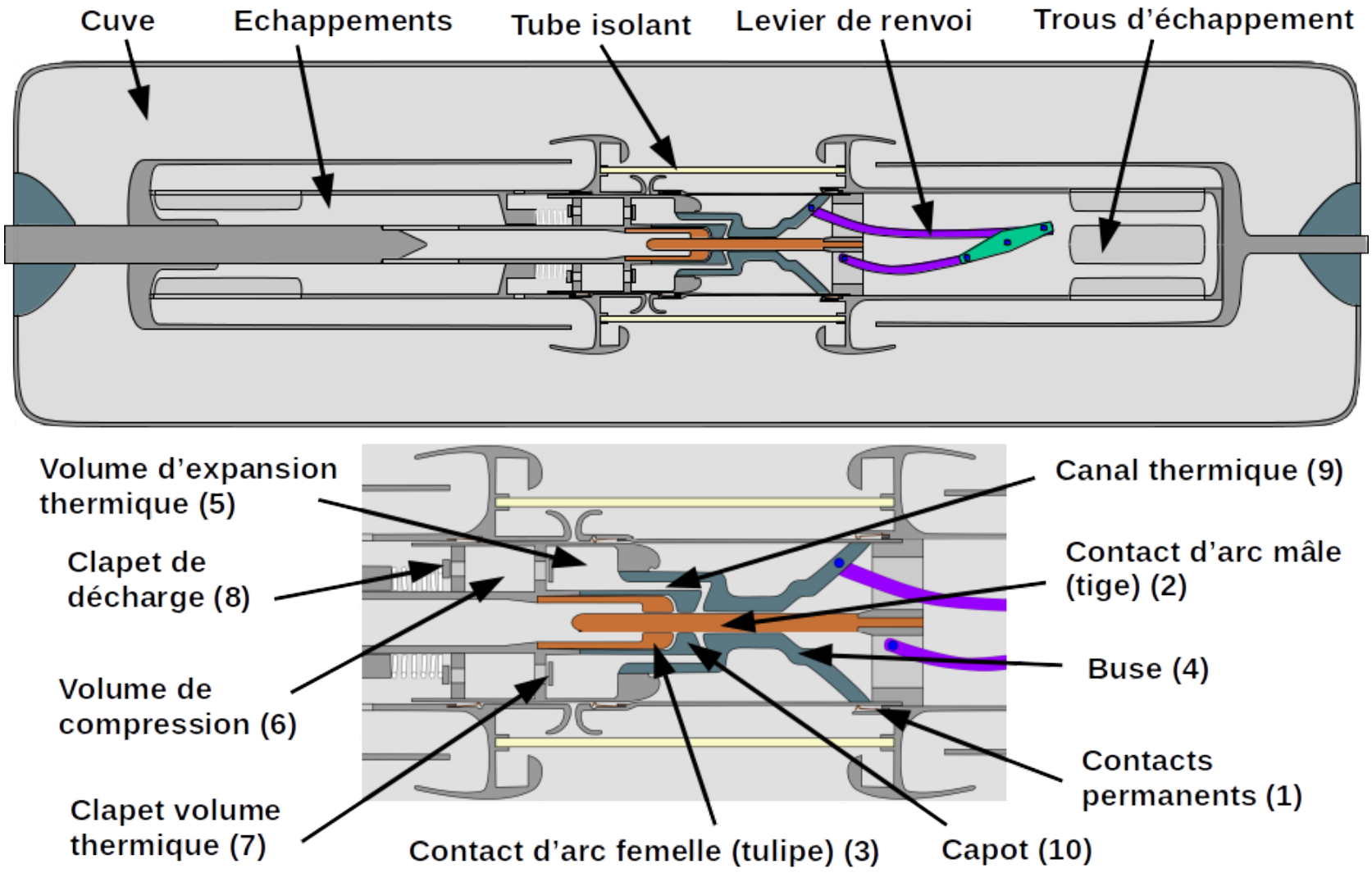


Figure 2.1 Architecture simplifiée d'un disjoncteur, illustrant sa complexité géométrique

Dans la plupart de ces appareils, on retrouve les composants suivants :

(1) **Contacts permanents** :
Pendant le fonctionnement normal le courant passe à travers ces contacts qui sont optimisés pour limiter les pertes par résistance de contact.

(2) **Contact d'arc mâle (tige)** :
C'est l'un des deux contacts mobiles dimensionnés pour supporter l'arc électrique qui s'y attache.

(3) **Contact d'arc femelle (tulipe)** :
La tulipe est le deuxième contact électrique mobile dédié à l'attachement de l'arc qui survient lors de la séparation avec la tige.

(4) **Buse** :
C'est un élément composé de PTFE et qui sert à contrôler l'écoulement de gaz pendant le mouvement de l'appareil. Son deuxième rôle est de générer des gaz supplémentaires par sublimation sous l'effet du rayonnement intense de l'arc, ce qui améliore l'efficacité du soufflage.

(5) **Volume d'expansion thermique** :
L'arc électrique apporte un gain d'énergie important au gaz qui rentre dans le volume thermique, cette accumulation de gaz chaud et pressurisé provoque un effet de soufflage sur l'arc lui-même lorsque celui-ci perd en puissance.

(6) **Volume de compression** :
Grâce aux mouvements des composants, ce volume se réduit, comprimant ainsi le gaz qui s'y trouve. En se vidant, le volume participe au soufflage de l'arc.

(7) **Clapet volume thermique** :
Il permet la communication de gaz du volume de compression vers le volume thermique et interdit l'écoulement inverse en se fermant lorsque la pression du volume thermique est supérieure à celle du volume de compression.

(8) **Clapet de décharge** :
Afin de limiter la pression du volume de compression à l'ouverture (pour réduire l'effort mécanique nécessaire) ce clapet s'ouvre à pression prédéterminée.

(9) **Canal thermique** :
Le gaz chaud et pressurisé produit par l'arc est guidé dans le volume d'expansion thermique à travers le canal thermique.

(10) **Capot** :
Le capot fait office de barrière diélectrique et thermique afin de protéger la tulipe.

2.1.2 Fonctionnement détaillé

La coupure par un disjoncteur à haute tension est obtenue en séparant les contacts d'arc dans un gaz isolant qui le plus souvent est du SF6. Lorsqu'un défaut est détecté, une commande mécanique est actionnée, provoquant le mouvement de certains composants de l'appareil. Dans un premier temps, il y a séparation des contacts permanents qui assurent le cheminement du courant en phase de fonctionnement normal, tel qu'illustré à la figure 2.2. Le courant transite désormais à travers les contacts d'arc :

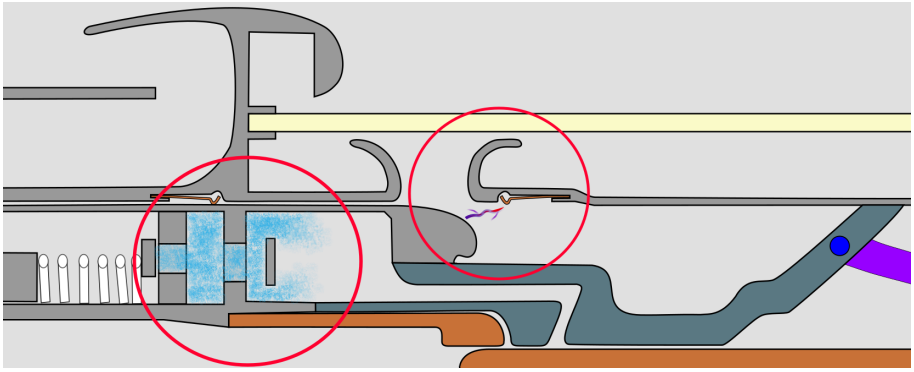


Figure 2.2 Séparation des contacts permanents

Avec le mouvement qui se poursuit, c'est au tour des contacts d'arc de se séparer. L'arc électrique se forme et assure le passage du courant (Figure 2.3) :

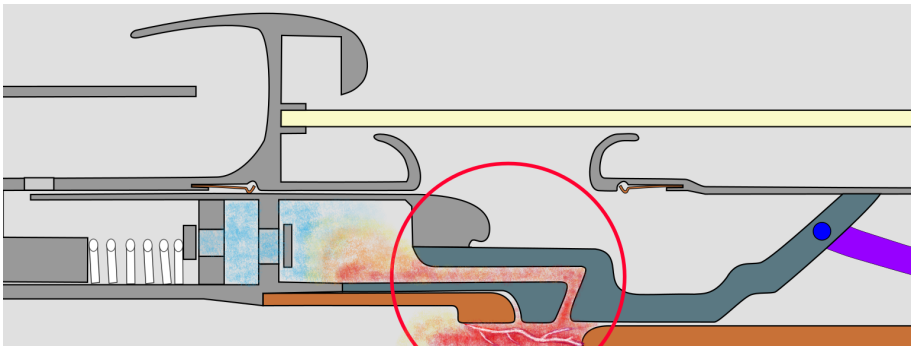


Figure 2.3 Séparation des contacts d'arc et allumage

Le passage du courant décompose le SF6 et le porte à une température avoisinant les 25000K.

Le volume thermique se pressurise et se remplit sous l'effet de la puissance apportée par l'arc, Figure 2.4.

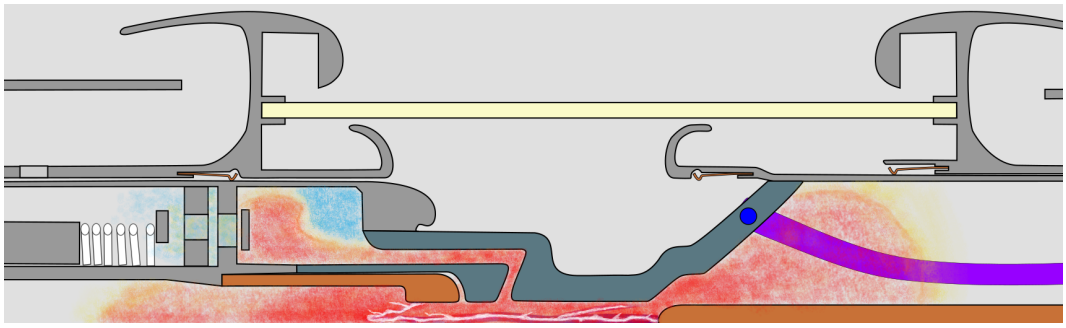


Figure 2.4 Chauffage et montée en pression du volume d'expansion thermique

L'onde de courant alternatif, généralement à 50Hz ou 60Hz, apporte une énergie qui varie et induit donc une source de pression (l'arc lui-même) qui varie en intensité au cours du temps. Au moment où l'onde approche un zéro de courant, il y a plus de pression dans le volume d'expansion thermique que dans l'arc, ce qui inverse l'écoulement et les gaz contenus dans le volume thermique se vident tout en soufflant sur l'arc. Ainsi, on met à profit le passage par zéro de l'onde du courant pour faciliter l'extinction de l'arc. En effet, en plus des gradients de pression qui produisent le soufflage, la puissance due à l'effet Joule diminue à cet instant, le milieu a alors tendance à se refroidir et donc passe d'un état conducteur à un état isolant. Si le soufflage a été suffisamment efficace (Figure 2.5), la coupure est réussie.

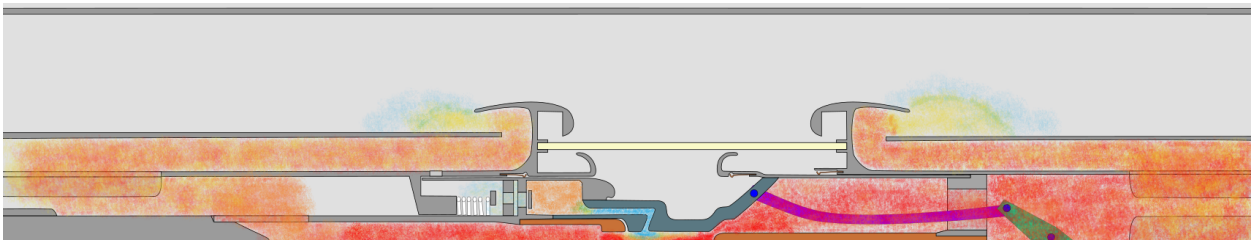


Figure 2.5 Soufflage et extinction de l'arc

Cependant, pour éviter un ré-allumage de l'arc aucun retour de gaz chaud ne doit avoir lieu, car la capacité d'isolement électrique du gaz dépend de sa pression, de sa température et de la distance physique entre les électrodes. Pour réussir la coupure dans les configurations prévues par les normes (IEC62271-100 ou ANSI-C3706/C3709 suivant les marchés) et probables dans les conditions d'opérations, les formes géométriques doivent être optimisées d'un point de vue

des écoulements fluides. Dans ces écoulements, les vitesses sont couramment de l'ordre de Mach 1.5, on veut contrôler les ondes de chocs et ne pas créer de dépressions dans des zones critiques. Après coupure, la tension aux bornes du disjoncteur se rétablit à la tension du réseau (Figure 2.6), provoquant d'importants champs électriques, ce qui accentue fortement le risque de ré-allumage.

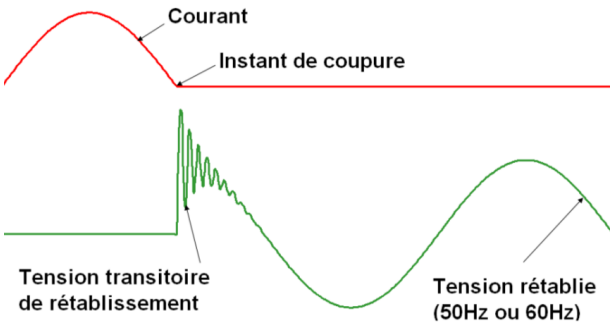


Figure 2.6 Tension transitoire de rétablissement après coupure (`fr.wikipedia.org`)

Il faut optimiser les formes géométriques en fonction de critères diélectriques (Figure 2.7), pour limiter les concentrations du champ électrique et les effets de pointe (effet pare-à-foudre à éviter). L'usure des appareils est un autre défi de conception car sous l'effet du rayonnement intense de l'arc (25000K), la buse est sublimée et l'attachement de l'arc sur les contacts (à base de Tungstène) les érode fortement.

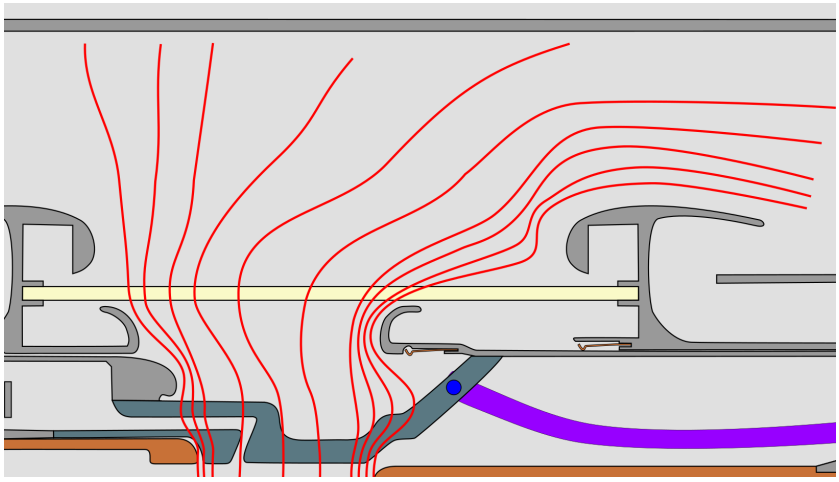


Figure 2.7 Isolignes de champ électrique après coupure

En résumé, les concepteurs font face à des phénomènes physiques très variés, et dont le comportement est fortement non-linéaire et couplé :

→ **Écoulement compressible supersonique :**
L'interruption du courant a lieu en quelques dizaines de milli-secondes, les pièces mobiles génèrent de forts gradients de vitesse/pression en plus des effets dus à l'arc.

→ **Champ magnétique :**
Le courant de court circuit peut atteindre plusieurs centaines de kilo-Ampères, le champ magnétique auto-induit qui en dépend est donc très important. Les forces magnétiques réduisent le diamètre de l'arc, ce qui concentre l'énergie dans une zone plus restreinte (la température est augmentée).

→ **Effet Joule :**
L'arc électrique se comporte comme une résistance électrique variable en fonction des conditions physiques environnantes (écoulement fluide, dissociation moléculaire, ionisation...). Le passage du courant échauffe donc le plasma.

→ **Rayonnement :**
La température du plasma est très importante et le rayonnement thermique est un phénomène physique majeur dans l'interruption de courant de court-circuit. Il participe à l'équilibre thermique en refroidissant la zone chauffée par l'effet Joule, c'est donc un mécanisme d'évacuation de l'énergie.

→ **Ablation :**
L'ablation est principalement due au rayonnement dans les longueurs d'ondes de l'ultra-violet. Les transferts de chaleur par convection peuvent aussi y contribuer.

→ **Champ électrique :**
Le rétablissement de la tension aux bornes du disjoncteur (après l'extinction) et l'écoulement fluide qui se poursuit, deviennent les phénomènes principaux à étudier.

Afin de faciliter la conception de ces appareils, les développements d'outils de calcul scientifique adaptés deviennent indispensables. Dans la prochaine section, nous aborderons les modèles numériques utilisés pour la simulation de l'opération des disjoncteurs haute-tension.

2.2 Modéliser les disjoncteurs

2.2.1 Le logiciel MC^3

Depuis près de 30 ans, le groupe GRMIAO a développé une approche spécialisée et originale pour la modélisation et la simulation des phénomènes présents lors de l'opération d'un disjoncteur haute-tension. De manière générale, la tâche consiste à représenter l'interaction arc-écoulement en combinant des modèles physiques spécifiques aux différents phénomènes mis en jeu.

Pour représenter l'interaction arc-fluide, les équations de conservation de la masse, de la quantité de mouvement et de l'énergie ont été utilisées car la colonne d'arc peut être considérée en équilibre thermodynamique local (ETL), c'est à dire que dans un petit volume, toutes les particules qui composent le milieu sont à la même température. L'utilisation d'une approche 2D (r-z) dans le logiciel MC^3 a été permise par le fait que le cœur de l'appareil, c'est à dire la région où l'arc s'allume peut être bien approximé en axisymétrique. En tant que pionnier dans le domaine, le groupe GRMIAO de l'École Polytechnique de Montréal, a tout d'abord généralisé le schéma de Roe [106] pour le calcul des écoulements compressibles sur les maillages mobiles arbitraires Eulériens-Lagrangiens (Trépanier et al. [122], Zhang et al. [131]). En effet, les vitesses caractéristiques du fluide sont de l'ordre de Mach 1.5 et les fortes variations de masse volumique dues également en partie à l'arc génèrent un écoulement fortement compressible. De plus, l'hypothèse d'un fluide non visqueux a donc été considérée puisque les phénomènes visqueux sont secondaires, tout du moins pendant la phase de fort courant. Ce sont donc les équations d'Euler qui sont utilisées avec l'adjonction de termes sources générés par l'arc. Les effets dominants sont : l'effet Joule dû au passage du courant et le rayonnement émis et absorbé par le gaz qui influencent le bilan d'énergie ; les forces de Laplace dues au champ magnétique produit par l'arc lui-même qui induisent une influence sur le bilan de quantité de mouvement. Le groupe GRMIAO a par la suite été parmi les premiers à développer des techniques pour la gestion des maillages de triangles mobiles et adaptatifs (Paraschivoiu et al. [86], Ilinca et al. [43], Trépanier et al.[123]). La combinaison d'un schéma numérique arbitrairement Eulérien-Lagrangien et d'une gestion de maillages mobiles permet de prendre en considération de manière conservative le mouvement des parois. Aussi, le caractère adaptatif du remaillage permet de capter les phénomènes physiques s'opérant à différentes échelles. Un extrait de maillage typique de géométrie de disjoncteur est présenté à la figure 2.8, où l'on peut voir que la complexité des géométries est importante. Les méthodes utilisées pour déplacer et adapter le maillage au fur et à mesure des déplacements des parois doivent être autant conservatives que possible d'un point de vue numérique et suffisamment robustes pour maintenir le maillage valide après mise à jour. C'est le cas avec la méthode

dite de raffinement-déraffinement utilisée pour l'adaptation ainsi que pour l'algorithme de déplacement de mailles qui combine judicieusement les vitesses de propagation des parois et le glissement des nœuds sur celles-ci.

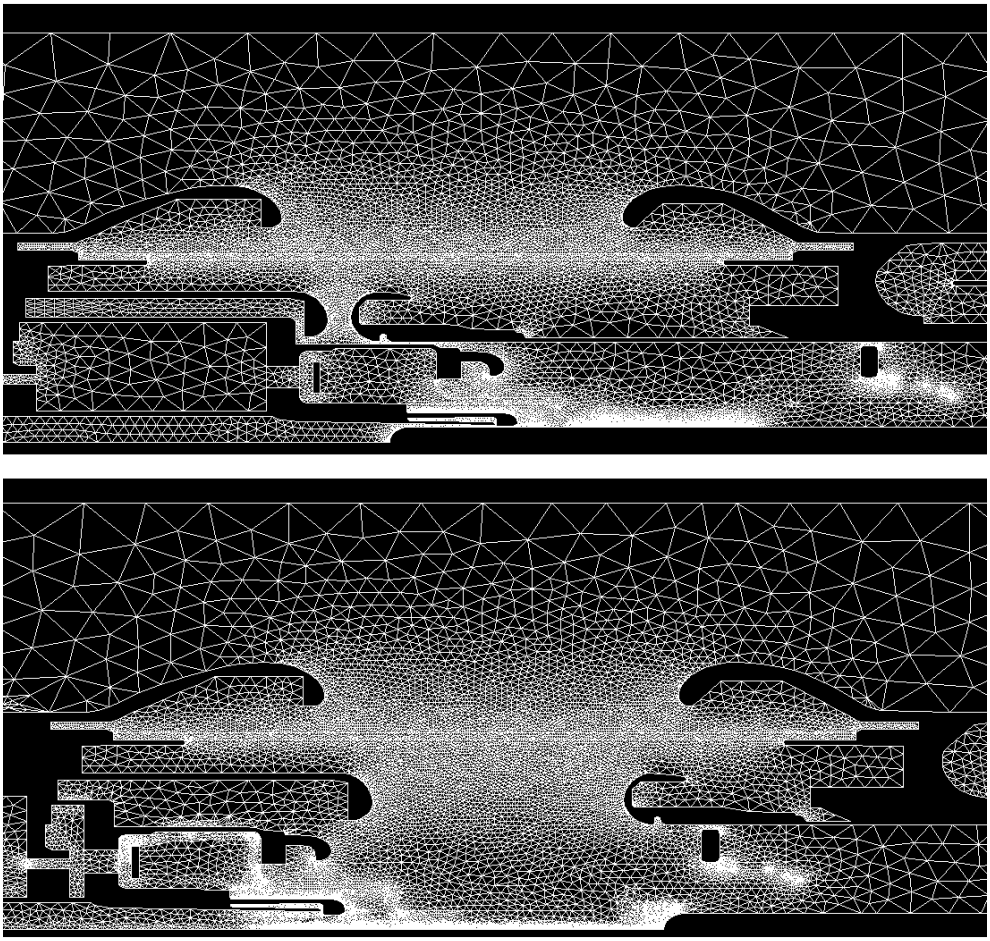


Figure 2.8 Maillage typique du coeur d'un disjoncteur - avant puis après déplacement

L'intégration de l'ensemble de ces techniques a constitué la base d'un code de calcul des écoulements compressibles instationnaires dans des géométries ayant des pièces mobiles. Sur ce noyau, d'autres modèles physiques ont été greffés afin de rendre l'outil adapté à la simulation d'arc électrique. Du fait des propriétés thermochimiques du SF6 et des conditions d'écoulement (vitesses élevées, fortes variations de pression, de l'ordre 5 à 100 bars) et compte tenu des grands écarts de la température qui peut varier de 300K dans certaines régions du fluide à plus de 25000 K dans les régions d'arc (l'état de la matière considéré est le plasma chaud pour ces régions), les hypothèses de gaz parfait ne sont plus valides. Les propriétés de gaz réels ont donc été appliquées par Godin et al. [33, 35]. Des modèles plus spécifiques aux arcs électriques ont alors été développés par Zhang et al. [132, 133]. La prise en compte des

équations de Maxwell a été introduite par Pellegrin et al. [91] pour prendre en compte les calculs du champ magnétique qui contracte l'arc notamment près des électrodes, la fréquence du courant (50Hz ou 60Hz) étant faible, les composantes instationnaires des équations de Maxwell ont été négligées. Le rayonnement thermique est le phénomène thermique dominant qui intervient lors de l'interruption d'un courant de court-circuit et des modèles spécifiques ont été introduits en adaptant la méthode des harmoniques sphériques (P-N) aux arcs électriques, notamment par Eby et al. [25]. Plus récemment la méthode des volumes finis (FVM) a été introduite par Melot et al. [76]. Cette dernière méthode est beaucoup plus coûteuse en temps de calcul (dans MC^3, environ 100 fois plus coûteuse que la méthode P1 sur les cas de disjoncteurs) mais elle est toutefois plus représentative de la physique du rayonnement. En effet, le modèle de type P-N correspond à un comportement elliptique, en contradiction avec le phénomène physique du rayonnement qui est un phénomène de propagation. Cela introduit dans les simulations numériques une diffusion thermique non physique. De plus, la prise en compte de conditions aux limites réalistes vis-à-vis de la physique est délicate avec le modèle P1. La méthode des volumes finis se base sur un lancer de rayons ce qui la rend par nature plus prédictive. Sous l'effet de l'intense rayonnement, les matériaux à base de PTFE qui constituent la buse sont sublimés. Des modèles ont alors été développés par Godin et al. [36], Martin et al. [70, 69] pour prendre en compte cet apport massique. L'ablation génère des gaz, principalement à base de C2F4 et autres produits issus de sa décomposition, qui sont introduits (fig 2.9) dans le système d'équation par l'intermédiaire de termes sources en énergie, en masse et en quantité de mouvement. Du fait de l'introduction de nouvelles espèces chimiques dans le milieu de remplissage, les équations de résolution des écoulements gazeux ont été étendues aux calculs multi-espèces par Martin et al. [70] et Arabi et al. [7]. L'outil scientifique MC^3 constitue ainsi un puissant outil pour le développement de produit, avec tous les requis de fiabilité et de robustesse que cela comporte (Robin-Jouan [105]). La figure 2.10 illustre les résultats de simulation que l'on peut obtenir à un instant donné de la phase d'arc. L'analyse de ces résultats permet de concevoir géométriquement la forme des éléments constituant l'ensemble des appareils de coupure.

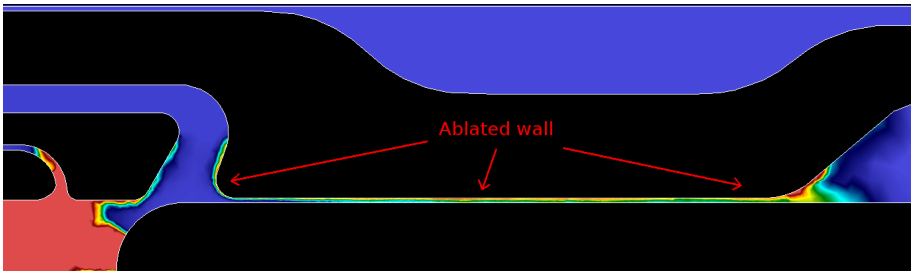


Figure 2.9 Extrait d'un résultat CFD typique - génération de gaz lors de l'ablation

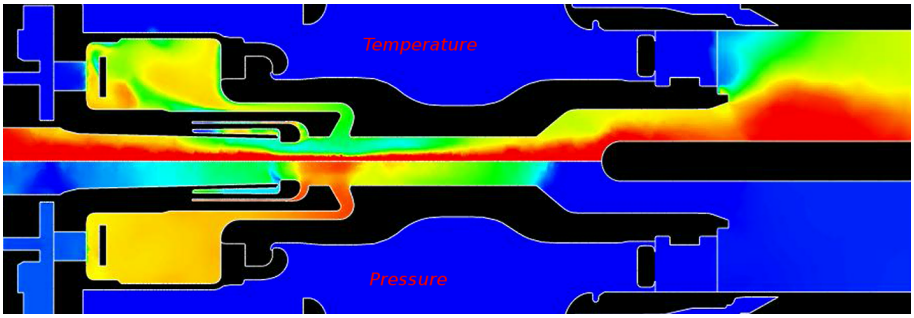


Figure 2.10 Extrait d'un résultat CFD typique - soufflage de l'arc

L'analyse et la comparaison des courbes de pression du volume thermique, mesurée en essais et calculée par MC^3, montrent une bonne corrélation (fig 2.11), contribuant à la confirmation d'une bonne prédiction du calcul pour la phase de fort courant.

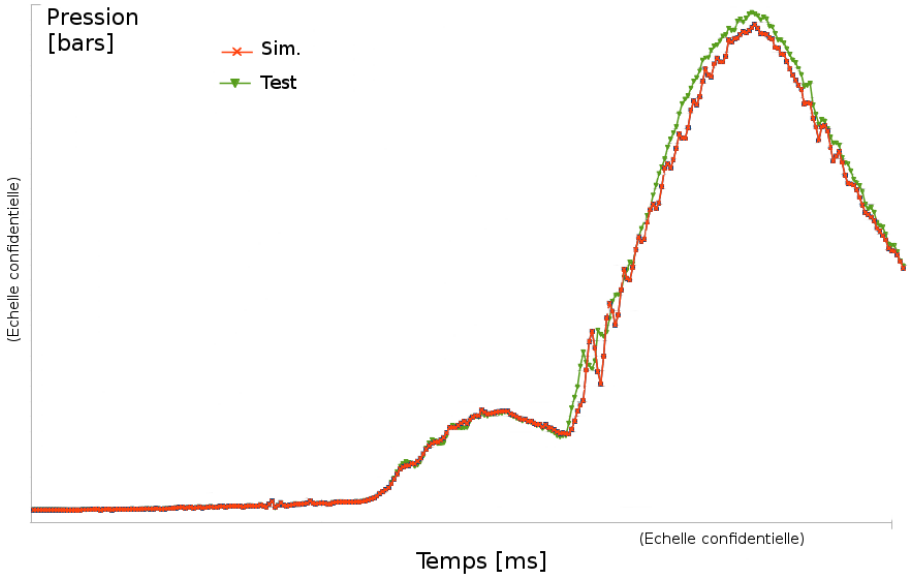


Figure 2.11 Exemple de comparaison de l'évolution de la pression - simulation vs essais

2.2.2 Les limites de MC^3

Toutefois, des extensions et des ajouts sont nécessaires pour repousser les limites de la modélisation actuelle, tant du point de vue de l'efficacité du calcul (qui peut atteindre pour certains cas complexes trois semaines de calcul, limitant de fait les capacités d'analyse et les possibilités de configurations d'étude) que du point de vue de la modélisation physique en vue d'améliorer la fidélité des modèles. Il a été démontré que l'hypothèse d'équilibre thermodynamique local (ETL) n'est plus valide près des électrodes et lors de la phase d'extinction.

Pour étudier ces effets, Girard et al. [32, 31] ont développé des modèles à deux températures où les plus petites particules, tels les électrons, sont à plus haute température que les éléments plus gros. Jusqu'à présent, ces modèles demeurent assez peu répandus en raison de leur complexité et en raison de leur faible influence sur les résultats globaux de la coupure. Plus récemment de nouveaux modèles sont apparus pour traiter également du non-équilibre chimique local, et les travaux de Tanaka et al. [117] sont particulièrement complets. En effet, au sein de l'arc les molécules sont dissociées par la forte température due au passage du courant. Ainsi, les éléments dissociés se recombinent, subissant ainsi de multiples réactions chimiques (D'après Girard et Gleizes [31], on peut en dénombrer 66 avec la mise en relation de 19 espèces différentes). La figure 2.12 montre la répartition des espèces à la pression de 0.1MPa dans un plasma de SF6 en fonction de la température, ces compositions de plasma se complexifient davantage avec l'ajout des espèces dues à l'ablation des parties en Teflon (PTFE) et des parties métalliques comme les contacts d'arc. Au moment de l'extinction, ces phénomènes thermo-chimiques ne sont plus négligeables. Avec le refroidissement du plasma, les ions et les électrons se recombinent pour donner des particules électriquement neutres, réduisant ainsi la conductance de l'arc.

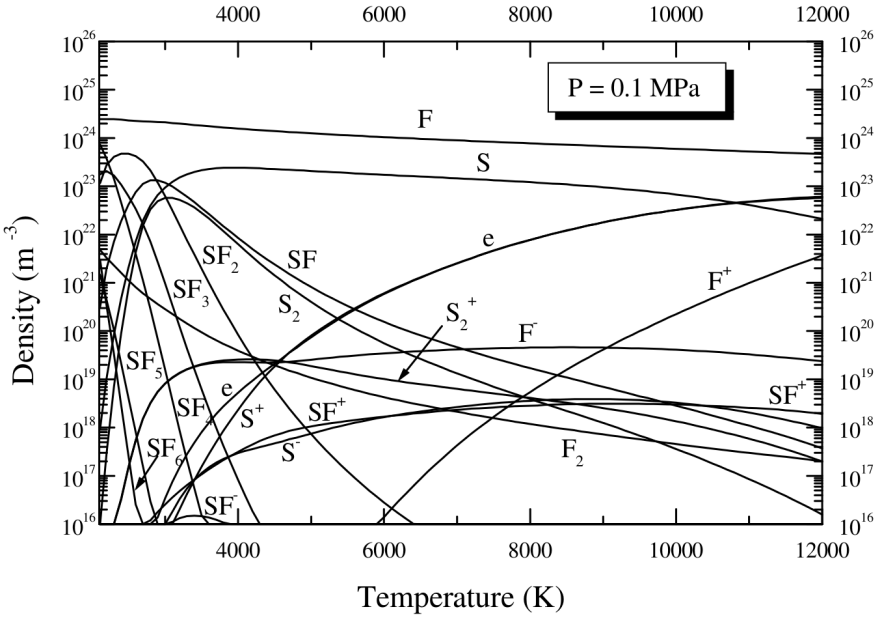


Figure 2.12 Composition du plasma de SF6 à l'équilibre - tiré de Girard et al.[31]

Cependant, les phénomènes physiques intervenant lors de l'extinction sont encore très mal connus et des modèles numériques spécifiques à l'extinction devront être développés, pour prendre en compte notamment les phénomènes d'instabilité 3D de l'arc au moment de l'extinction. Ces instabilités s'apparentent à de la turbulence, aussi des auteurs ont tenté d'utiliser

des modèles de turbulence classiques, comme $k-\varepsilon$ ou à longueur de mélange. Ce dernier semble donner de meilleurs résultats que $k-\varepsilon$, d'après Yan et al. [129], Song et al. [51] et Seeger et al. [12]. D'autre part, Bini et al. [12] ont montré avec succès que pour simuler les écoulements dans les échappements, l'utilisation du modèle de turbulence $k-\varepsilon$ était satisfaisante. Néanmoins, il est aujourd'hui clair qu'un modèle spécifique aux arcs électriques doit être développé, et comme il a déjà été montré par Gleizes et al. [34], l'interaction de l'arc couplé avec le circuit électrique doit être prise en compte pour capter les phénomènes électriques transitoires. D'autres phénomènes physiques, comme l'interaction arc-électrodes ont été étudiés (Maruzewski et al. [71], Benilov [10, 50], Chévrier et al. [50]), mais le phénomène est très complexe à cause de sa nature multi-physiques (fluide, électrique, magnétique, vapeurs métalliques, interaction avec le circuit...). Mises à part les instabilités de l'arc au moment de l'extinction et les interactions arc-électrodes, d'autres phénomènes 3D interviennent dans les disjoncteurs haute-tension. Les écoulements à travers des clapets de décharge ou anti-retour et les écoulements dans les échappements en font partie, les géométries étant bien moins judicieusement approximées en axisymétrique. De plus, dans ces régions, les effets visqueux ne sont plus négligeables car les échelles de temps et d'espace sont différentes. La résolution 3D complète demeure inaccessible à cause de la taille des résolutions pour la structure de l'écoulement, des complexités géométriques et des multiples échelles en jeu. La recherche sur la modélisation des phénomènes post-arc est également très active. En effet, après extinction, si le milieu reste trop chaud dans l'espace inter-électrodes et du fait de la différence de potentiel croissante entre les électrodes durant la phase transitoire de rétablissement de la tension du réseau aux bornes du disjoncteur, des décharges électriques peuvent apparaître et causer le réallumage de l'arc. La prédiction de la réussite ou de l'échec de la coupure de l'arc électrique et de ses risques de réallumage peut se faire en introduisant des critères de coupure, qui au-delà d'une valeur seuil déterminent les chances de réussite complète de la coupure. Certains auteurs ont proposé une méthode basée sur les champs électriques locaux et la densité locale de particules gazeuses (Pedersen [89], Trépanier et al. [121]). C'est une méthode intuitive puisque la capacité d'isolement du gaz sous un certain champ électrique dépend directement de sa densité. Cependant, cette méthode repose sur une valeur seuil de référence à établir expérimentalement et qui, pour des calculs d'arc, s'est révélée difficile à déterminer en pratique, voire impossible, rendant ce critère non prédictif. Une nouvelle méthode a été introduite par Stoller, Seeger et al. [116, 114] pour prendre en compte l'énergie (thermique et électrique) disponible localement et la comparer à l'énergie totale nécessaire à l'amorçage d'une décharge (théorie du streamer-leader). D'un point de vue pratique, cette méthode se révèle plus prédictive quant à la réussite de la coupure mais son évaluation reste coûteuse en temps de calcul.

2.2.3 Vers une nouvelle version de MC^3

Tous ces phénomènes physiques liés à la modélisation des disjoncteurs haute-tension sont fortement multi-physiques et multi-échelles. Multi-échelles, ils le sont, que ce soit d'un point vue temporel (par exemple, les temps caractéristiques des écoulements fluides et des recombinaisons chimiques sont radicalement différents) que d'un point de vue spatial (les phénomènes d'interactions arc-électrodes sont très localisés (quelques millimètres cubes), tandis que les tourbillons de gaz intervenant dans les échappements sont beaucoup plus étendus (quelques milliers de millimètres cubes)). De plus, d'un point de vue purement spatial les phénomènes qui se combinent sont assimilables soit à des comportements 2D axisymétriques soit à des comportements 3D. La conception architecturale du logiciel MC^3 et les modèles intégrés ne le prennent pas en compte.

D'un point de vue de la modélisation des disjoncteurs haute-tension, le couplage entre le logiciel ANSYS-CFX et MC^3 a été exploré par Petro et Trépanier [92]. Cependant, un échange uniquement unidirectionnel de conditions aux limites de MC^3 (2D-axisymétrique) vers ANSYS-CFX (3D) était fait via la lecture des fichiers d'échange, ce qui était très peu efficace en terme de temps de calcul et de précision. De plus, cela supposait de connaître le sens de circulation des gaz à priori. Cependant, par ses résultats numériques, l'étude a contribué à démontrer que le potentiel d'un couplage 2D-3D était intéressant pour l'application aux simulations des disjoncteurs haute-tension.

Dans la prochaine section, on étudie quelques principes de modélisation multi-physiques et de méthodologies de couplages multi-physiques et multi-échelles existantes et on montre les possibilités, les avantages et les limites à considérer dans un contexte de modélisation des disjoncteurs haute-tension. Ceci nous permettra de proposer des améliorations architecturales pertinentes, et d'introduire une méthodologie de couplage 2D-3D plus efficace.

2.3 La modélisation multi-physiques et multi-échelles et les méthodologies de couplage

2.3.1 Définitions et classifications

Au cours des dernières années, quelques auteurs ont proposé des classifications et des définitions propres au domaine des problèmes multi-physiques et multi-échelles. Larson [63] propose une définition d'un modèle couplé, qui consiste à résoudre deux ou plus, systèmes d'équations (ou constituants), en général couplés. Larson [63] propose aussi de représenter les systèmes couplés via une approche empruntée à la théorie des graphes où des nœuds représentent les différents constituants et des flèches qui les relient sont utilisées pour représenter

leurs interactions.
Récemment, Keyes et al. [53] proposent une revue des défis associés aux problèmes multi-physiques et multi-échelles. Ils identifient clairement divers types de problèmes, qui peuvent être classifiés comme multi-physiques et multi-échelles.
Nous proposons de représenter la classification des problèmes multi-physiques et multi-échelles dans un espace de couplage présenté à la figure 2.13. Cet espace comporte trois axes : spatial, temporel et fidélité, ou échelle de modélisation. Cette figure sera reprise en fin de section pour décrire les modèles multi-physiques et multi-échelles dans les disjoncteurs.

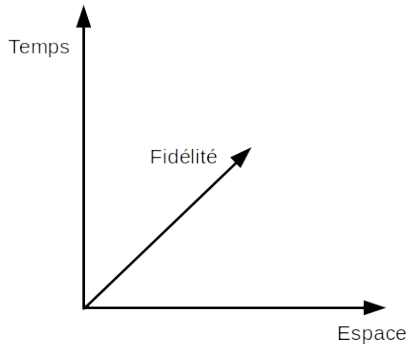


Figure 2.13 Espace de couplage

2.3.2 Couplages spatiaux

Ce type de couplage implique deux points dans l'échelle d'espace, qui sont situés à différents endroits dans l'espace de couplage. Le couplage spatial peut être volumique (par l'adjonction de termes sources,...) ou surfacique (par l'intermédiaire d'interfaces, de forces surfaciques...). L'échange d'informations peut être soit unidirectionnel soit bidirectionnel. Les physiques du couplage peuvent être couplées de manière plus ou moins forte. Par exemple, on peut choisir de résoudre un des systèmes d'équations avant l'autre et d'introduire le résultat comme point d'entrée dans le système d'équations suivant, le couplage est alors plutôt faible. Si au contraire on choisit de coupler les systèmes d'équations et de réaliser des sous-itérations jusqu'à convergence simultanée ou bien de les coupler implicitement lors d'une intégration temporelle, alors le couplage est d'un niveau plus fort. Au niveau du couplage spatial, les méthodes doivent prendre en compte les aspects du transfert entre maillages tout en étant autant conservatives que possible, les méthodes utilisées sont la plupart du temps des méthodes interpolantes ou chimère/overset (par exemple, Ryan et al. [110], Miller et al. [77] pour le couplage fluide-structure, Swartz et al. [113] proposent le couplage de deux solveurs fluides avec discrétisation spatiale et temporelle arbitraire, ou Wang et al. [126] qui autorisent même

la prise en compte de trous et proposent un algorithme pour faciliter les communications inter-grilles).

2.3.3 Couplages multi-fidélités

Les couplages multi-échelles (appelés aussi multi-fidélités), dont font par exemple partie la modélisation du climat (en plus du caractère multi-physiques, Larson [61]) ou la modélisation de la propagation de fissure dans les solides comme proposé par Chen et al. [20], peuvent être vus comme une forme particulière de couplage multi-physiques, où les différents systèmes d'équations représentent le même système à différentes échelles, habituellement spatiales. Lorsque le transfert d'information se fait d'une échelle grossière vers fine, on parle d'élévation tandis que dans le sens inverse, on parte plutôt de restriction de l'information (Keyes et al. [53]). S'il s'agit de la même physique mais calculée sur différents maillages alors le terme plutôt employé est multi-résolutions.

2.3.4 Couplages temporels

Si le couplage multi-échelles est temporel alors on parle de multi-taux (comme par exemple dans le couplage des phénomènes de combustion). Il peut être effectué dans différentes régions et on a alors un pas de temps par région ou bien il peut être effectué dans la même région en multi-échelles et on adopte les techniques d'élévation et/ou de restriction. De plus, la résolution temporelle peut être couplée explicitement (ce qui est le plus utilisé dans les applications multi-physiques du fait de sa simplicité) ou implicitement (très stable du fait de l'élimination des conditions de stabilité sur le pas de temps par exemple, mais plus difficile à mettre en œuvre dans les contextes multi-physiques).

2.3.5 Approches de résolution

Les approches pour résoudre le couplage dépendent du type de problème. Il peut s'agir de problèmes d'équilibre ou bien de problèmes d'évolution (en temps) et dans ce cas des approches ont été introduites comme le "loose-coupling" et autres variantes. Cependant, Keyes et al.[53] suggèrent d'adopter des méthodes de couplages plus serrées/fortes pour des questions de stabilité et de performances sur les futures machines de calcul.

De récents développements proposent d'employer des méthodes d'intégration temporelle hybrides entre explicite et implicite que l'on appelle méthodes IMEX (Patel et al. [88], Vermeire et al. [124], Weller et al. [127], Lemieux et al. [64]), où l'on résout explicitement dans certaines régions et implicitement dans d'autres. Ces nouvelles méthodes proposent un com-

promis entre simplicité de mise en œuvre, scalabilité (performance) et stabilité. On combine ainsi les avantages des deux méthodes en essayant d'éviter leurs inconvénients. Afin de prédire les choix pratiques optimaux, certains auteurs ont proposé des modèles de prédiction et des algorithmes pour développer efficacement un couplage IMEX (Rouson et al. [108], Larson et al.[62], Kim et al. [55, 56], Weinan et al. [2], Patel et al.[88], Lemieux et al.[64]). Afin d'adapter le couplage à de multiples problèmes, certains auteurs proposent un certain nombre de cadres ou de boites à outils spécialisés en fonction de leur propre classification multi-physiques, comme par exemple COOLFluid de Lani et al. [59], Larson et al. [62], MUSE framework de Zwart et al. [135], Mishra [78], Tautges et al. [118], Cary et al. [18].

2.3.6 Architecture logicielle

En ce qui concerne le développement d'une architecture logicielle adéquate et d'un couplage optimal, Keyes et al. [53], Larson [63], Kim et al. [55, 56] soulèvent des points importants que nous pouvons prendre en compte dans notre projet de recherche. Il faudra choisir un niveau d'encapsulation suffisant pour faciliter les tâches de maintenance, la réutilisation et la compréhension du code sans toutefois alourdir l'implémentation (pertes de performance). Comme les différents modèles partageront des informations, une infrastructure commune pour exploiter les points communs et réutiliser les méthodes et des portions de code communes entre les différents champs d'application devra être développée, tout en permettant de futures extensions à d'autres problèmes. L'implémentation doit cependant rester suffisamment évolutive et non restrictive pour également s'adapter aux futures évolutions des machines de calculs (calculs GPGPU, Co-processeurs...). Un avantage direct d'écrire des solveurs spécialisés est le gain de performance. En effet, cela permet la maîtrise complète de la chaîne d'implémentation/documentation/maintenance/test et de s'adapter aux matériels informatiques et bibliothèques de programmation que l'on envisage d'utiliser [44, 45, 46, 81, 82, 54]. Des choix plus pratiques peuvent également être considérés, par exemple la résolution de différentes physiques sur différents maillages peut conduire à une optimisation importante, dépendamment du coût de calcul associé à chacune d'entre elles et du coût de la génération et du maintien de plusieurs maillages (Keyes et al. [53]). Cependant, résoudre toutes les physiques sur un seul maillage qui est adapté/raffiné en fonction des besoins de chacune des physiques peut tout de même être un bon compromis, notamment si la complexité géométrique est importante et la génération de différents maillages trop critique à réaliser. De plus, l'utilisation de plusieurs maillages nécessite des interpolations fréquentes entre les maillages et les modèles, ce qui sacrifie une partie de la précision de calcul. Cependant, un moyen d'éviter les erreurs lors du transfert de données et du couplage multi-physiques est d'utiliser des maillages hiérarchiques comme l'ont proposé Solin et al. [115].

D'autres techniques de couplage permettent même de coupler des méthodes de discrétisation numérique différentes, par exemple les techniques SANS-maillage avec les méthodes classiques AVEC-maillage que proposent Fries et al. [29, 30]. Cela peut être avantageux lorsque la génération de maillage est trop difficile ou que la précision n'est pas le critère prioritaire dans une des régions ou pour une des physiques. Certains auteurs comme Halama et al. [28], Sardella [112] ou Wu et al. [128] combinent également la méthode des volumes finis avec la méthode des éléments finis, le but étant de tirer parti des avantages de l'une ou de l'autre des méthodes dans les situations adaptées.

2.4 Objectifs spécifiques de la thèse

Des limites actuelles de l'outil MC^3 (2.2.2) et des besoins d'extensions vers une nouvelle version de MC3 (2.2.3), se dessinent les objectifs spécifiques de la thèse. Le but principal de cette thèse est de rendre l'outil de simulation plus performant en terme de temps de calcul et d'améliorer la précision de ses résultats. En adéquation avec les trois axes de recherche introduits précédemment dans le chapitre d'introduction, on vise les objectifs spécifiques suivants :

1. **Choisir les modèles pertinents :**
   Le premier objectif est de confirmer ou infirmer les choix des modèles actuellement en place dans MC^3. Cette tâche s'appuie en partie sur la revue de la littérature précédemment critiquée et d'autres éléments de confirmation seront apportés en abordant le chapitre qui suit.
2. **Optimiser les modèles avec le support de l'analyse de MC^3 :**
   Le second objectif est d'analyser MC^3 de manière approfondie pour ensuite proposer des stratégies d'amélioration bénéfiques à la vitesse d'exécution et/ou à la précision des calculs. Les points d'analyse porteront sur les choix suivants : des structures de données pour le maillage et les quantités physiques ; des formats matriciels, des solveurs linéaires ; des algorithmes pour exposer au maximum le parallélisme et la scalabilité ; du style d'écriture du code source pour favoriser l'interprétation par le compilateur et donc l'optimisation de l'exécutable final.
3. **Proposer une architecture adéquate :**
   Les différents modèles physiques doivent être couplés de façon adaptée et exposer le parallélisme de tâches. Les opérations de maintenance représentent le facteur le plus important dans le coût financier global du logiciel. En effet, elles y contribuent avec une portion supérieure à 95 % (Pigoski [93]). La performance en temps de calcul, la lisibilité du code source et la facilitation des évolutions futures et de la maintenance

sont des points qui peuvent se révéler contradictoires. Le troisième objectif produira une architecture conçue pour présenter un équilibre entre ces contraintes.

4. **Améliorer la prédiction des simulations :**
   La précision des simulations numériques peut être améliorée par l'ajout des calculs de mécanique des fluides tri-dimensionnels, cela constituera notre quatrième objectif. Il s'agit du premier point d'amélioration à prendre en compte car cela influence directement la conception des appareils. Cependant, résoudre les écoulements fluides en 3D pour une géométrie complète s'avère inefficace et non-pertinent. À l'inverse, combiner des zones de calculs 3D en compléments de zones 2D axi-symétriques s'avère plus judicieux. Nous devrons donc développer une méthode de couplage entre les régions de différentes dimensions.

Ces objectifs peuvent être vus comme une modernisation du code de calcul scientifique sur les plans de la modélisation, de l'algorithmique et de l'adaptation de la programmation au matériel informatique dans le contexte du calcul haute performance appliqué aux simulations numériques des disjoncteurs haute-tension.

CHAPITRE 3 PERTINENCE DES MODÈLES DE MC3 ET DÉTAILS

3.1 Sélection des modèles multi-physiques et multi-échelles

Les différents modèles décrits à la section 2.2 peuvent être classifiés selon l'approche multi-physiques et multi-échelles, telle que présentée à la figure 3.1.

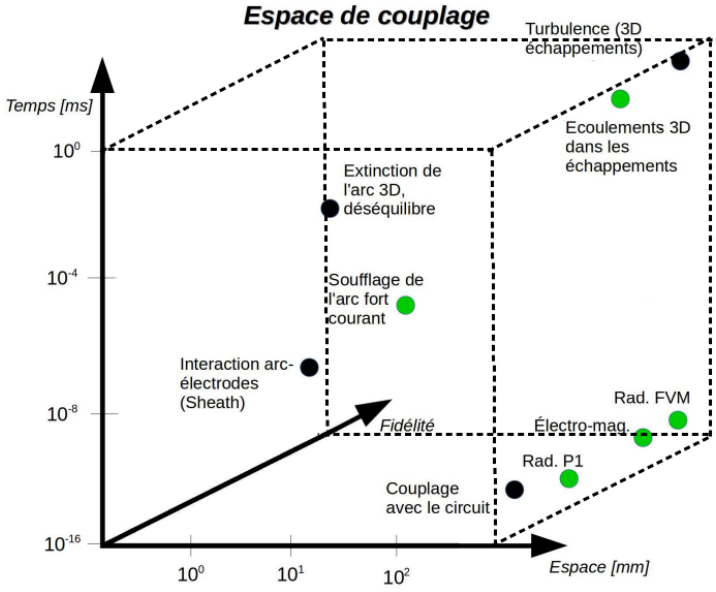


Figure 3.1 Classification des modèles pour le cas des disjoncteurs

Dans un premier temps, nous désirons nous pencher sur une intégration des modèles qui sont identifiés en vert sur la figure 3.1. Ces modèles physiques essentiels qui seront couplés et sur lesquels on se concentrera sont : le modèle des écoulements de fluides en gaz réels (Arabi et al. [7], Trépanier et al. [122], Zhang et al. [131], Roe [106]) ; le modèle pour le calcul des propriétés électriques (termes sources d'effet Joule liés à l'arc, champ électrique) de Zhang et al. [132, 133] ; le modèle pour la prise en compte du calcul des champs magnétiques de Pellegrin et al. [91] ; les modèles de rayonnement P1 de Eby [25] et FVM de Melot [76] et de l'ablation (Godin et al. [36], Maruzewski et al. [71] et Martin et al. [69]). Ils représentent les besoins actuels en terme d'application et sont les modèles les plus importants pour la conception des appareils. Les équations correspondantes sont décrites dans les sections qui suivent. On prendra soin de laisser le champ libre à la possibilité d'ajouts dans de futures révisions de l'outil scientifique d'autres modèles qui ne sont pas pour le moment vitaux ou non suffisamment éprouvés comme le déséquilibre thermodynamique proposé par Girard, Gleizes et al. [32, 31],

le déséquilibre chimique de Tanaka et al. [117], les modèles d'interaction arc-électrodes (Maruzewski et al. [71], Benilov [10], Chévrier [50]), le couplage avec le circuit d'après Gleizes et al. [34] et les modèles de turbulence (Yan et al. [129], Song et al. [51], Seeger et al. [12]).
Au cœur des besoins d'extension actuels, on a identifié le couplage des écoulements 3D à grande échelle d'espace et de temps dans les échappements avec les écoulements axi-symétriques au cœur de l'arc lors du soufflage à fort courant. Ceci demande de se concentrer sur deux aspects de ce couplage : le couplage 2D-axisymétrique-3D entre les deux modèles et le traitement des échelles temporelles différentes entre les deux régions. L'intégration du modèle de couplage devra être facilitée par une architecture logicielle que nous aurons préalablement conçue et adaptée à nos besoins fortement multi-physiques. Un modèle de couplage 2D-3D sera proposé au chapitre 6, un modèle de résolution des équations fluides en 3D y sera également introduit.

3.2 Systèmes d'équations et détails d'implémentation dans MC3

3.2.1 Solveur fluide

Les calculs d'écoulements instationnaires et compressibles, associés aux parois et discontinuités mobiles nécessitent un schéma numérique relié au déplacement du maillage. Cela requiert une séquence de maillages qui permet de décrire avec précision l'évolution de l'écoulement et du domaine de calcul. Le schéma numérique doit correctement prendre en compte le mouvement du maillage. Dans notre cas, une discrétisation du domaine de calcul par des triangles a été choisie, ce qui permet une très grande flexibilité en vue d'une adaptation du maillage. Le solveur approché de Riemann introduit par Roe [106] a été sélectionné pour réaliser les simulations numériques et a été étendu aux maillages mobiles par Trépanier et al. [122, 131].

Équations de base

Le système d'équations d'Euler décrivant un écoulement de fluide 2D axi-symétrique et non visqueux, dans l'espace cartésien (r, z), peut être écrit sous forme conservatrice telle que :

$$\frac{\partial \vec{U}}{\partial t} + \frac{1}{r}\frac{\partial \vec{F}_r}{\partial r} + \frac{\partial \vec{F}_z}{\partial z} = \vec{S} \tag{3.1}$$

où,

$$\vec{U} = \begin{bmatrix} \rho \\ \rho u_z \\ \rho u_r \\ \rho E \end{bmatrix}, \quad \vec{F}_r = \begin{bmatrix} r\rho u_r \\ r\rho u_z u_r \\ r(\rho u_r^2 + P) \\ r u_r(\rho E + P) \end{bmatrix}, \quad \vec{F}_z = \begin{bmatrix} \rho u_z \\ \rho u_z^2 + P \\ \rho u_z u_r \\ u_z(\rho E + P) \end{bmatrix} \; et$$

$$\vec{S} = \vec{S_{axi}} + \vec{S_{arc}} = \begin{bmatrix} 0 \\ 0 \\ P/r \\ 0 \end{bmatrix} + \begin{bmatrix} S_m \\ S_z \\ S_r \\ S_{ohm} + S_{rad} + S_{abl} \end{bmatrix}$$

avec ρ la masse volumique, P la pression, $\vec{V} = (u_r, u_z)^T$ le vecteur vitesse et $E = e + \frac{1}{2}||V||^2$ l'énergie totale où e est l'énergie interne.
Les termes sources d'arc $\vec{S_{arc}}$ seront décrits dans la section 3.2.2. Le système d'équation 3.1 est fermé en utilisant une équation d'état. Due à la dissociation et l'ionisation des molécules du gaz porté à haute température ($\approx$ 25 000 $Kelvin$), une équation d'état de type gaz réel doit être employée. L'état thermodynamique du gaz peut encore être caractérisé comme une fonction de deux variables thermodynamiques indépendantes, du fait de l'hypothèse d'ETL. Les propriétés thermodynamiques et les propriétés physiques du gaz en équilibre et du plasma sont obtenues de données publiées ou de tables de données générées par un laboratoire spécialisé en détermination des propriétés des gaz et des plasmas.

Intégration volumes finis

La forme intégrale du système d'Équations 3.1 utilisée par la méthode des volumes finis peut être écrite telle que :

$$\frac{\partial}{\partial t}\int_V \vec{U} dV + \oint_{\partial V} (\vec{F_r}.n_r + \vec{F_z}.n_z) dS = \int_V \vec{S} dV, \tag{3.2}$$

avec V le volume de contrôle, ∂V la frontière délimitant celui-ci et $\vec{n} = (n_r, n_z)$ le vecteur unitaire normal sortant de la frontière du volume de contrôle. Les variables conservatives $\vec{U}$ sont avancées du temps n au temps $n+1$ explicitement avec la méthode d'Euler :

$$U_i^{n+1} = U_i^n - \frac{\Delta t}{V}\sum_{c=1}^{3} \mathbf{F}_c^n S_c + \Delta t S_i^n, \tag{3.3}$$

où S_c est la surface des interfaces entre les volumes de contrôle triangulaires. Les flux $\mathbf{F}$ aux interfaces sont obtenus avec le schéma de Roe [106] en version gaz réel proposée par Glaister [33], Trépanier et al. [35] et plus récemment améliorée par Arabi et al. [7]. La base du schéma de Roe est l'introduction d'un état moyen $\tilde{\mathcal{A}}$ pour approximer la jacobienne $\mathcal{A} = \frac{\partial \mathbf{F}}{\partial \mathbf{U}}$ à chaque interface. La matrice approchée $\tilde{\mathcal{A}}$ peut s'exprimer sous la forme :

$$\tilde{\mathcal{A}} = \tilde{\mathcal{Q}}\tilde{\mathcal{D}}\tilde{\mathcal{Q}}^{-1},$$

où, pour le cas 2D, $\tilde{\mathcal{D}} = diag(\tilde{\lambda_1}, \tilde{\lambda_2}, \tilde{\lambda_3}, \tilde{\lambda_4})$, avec

$$\tilde{\lambda_1} = \tilde{u}_n + \tilde{c}, \quad \tilde{\lambda_2} = \tilde{u}_n, \quad \tilde{\lambda_3} = \tilde{u}_n, \quad \tilde{\lambda_4} = \tilde{u}_n - \tilde{c},$$

les quatre valeurs propres de $\tilde{\mathcal{A}}$, et $\mathcal{Q}$ qui est formé par les vecteurs propres indépendants correspondants.

Les flux à chaque interface avec les états droit et gauche représentés par R et L sont ensuite calculés avec :

$$\mathbf{F} = \frac{1}{2}(\mathbf{F}_R + \mathbf{F}_L) - \frac{1}{2}\tilde{\mathcal{Q}}|\tilde{\mathcal{D}}|\tilde{\mathcal{Q}}^{-1}(\mathbf{U}_R - \mathbf{U}_L). \tag{3.4}$$

Ces flux dépendent des états gauche et droit de l'interface (Figure 3.2), qui sont constants par cellule dans le cas d'un schéma au premier ordre ou reconstruits par une interpolation linéaire par morceau dans le cas d'un schéma du deuxième ordre.

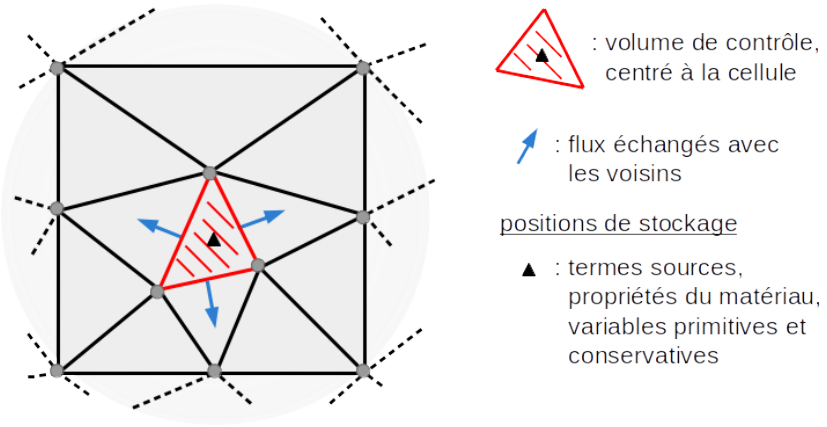


Figure 3.2 Échanges de flux entre cellules voisines

Conditions aux limites

Dans le cas des disjoncteurs haute-tension, les conditions aux limites sont de type : axe de symétrie, parois solides ou parois ablatées (avec injection de gaz). Les conditions aux limites sont imposées avec la technique des cellules miroirs (ou cellules fantômes). Dans le cas d'un axe de symétrie ou d'une paroi solide, la composante normale de la vitesse est réfléchie, tandis que les autres propriétés sont simplement copiées de la cellule intérieure vers la cellule fantôme. Aux parois ablatées, des injections de masse, d'énergie et de quantité de mouvement sont additionnées au bilan de la cellule intérieure. Ensuite, le solveur de Riemann est appliqué de la même façon à tous les côtés des volumes de contrôle quel que soit le type de chacun (interne, parois solide, ablatée, axe). Pour permettre une validation de l'outil de simulation plus aisée, des conditions aux limites supplémentaires à celles utilisées pour la simulation des

disjoncteurs, telles que les conditions d'entrées et de sorties ont été ajoutées, ce qui permet de réaliser des cas tests de validation non envisageables autrement.

3.2.2 Solveur de Helmholtz et termes sources associés

Formulation volumes finis

Une technique basée sur les volumes finis est également employée pour la résolution des équations elliptiques résultantes de la modélisation des divers phénomènes qui entrent en jeu dans les chambres de coupure des disjoncteurs. Ces phénomènes comprennent : les équations pour le champ électro-statique, pour le champ magnéto-statique et pour le rayonnement par la méthode P1. Ces équations seront rappelées dans cette section.

Les volumes de contrôle considérés sont de type triangle ainsi que leur covolume centré sur les nœuds (voir Figure 3.3). Les propriétés du matériau et les termes sources d'arc sont stockés au centre des triangles en relation directe avec le solveur fluide et pris pour constantes sur chacun d'eux. Tandis que, les variables inconnues sont stockées et résolues aux sommets des triangles en utilisant le covolume. Ce système de stockage hybride, très ressemblant à une formulation éléments finis, accepte naturellement les discontinuités dans les valeurs des termes sources et des propriétés du matériau aux interfaces des triangles tout en assurant une variation continue pour les valeurs des inconnues aux nœuds.

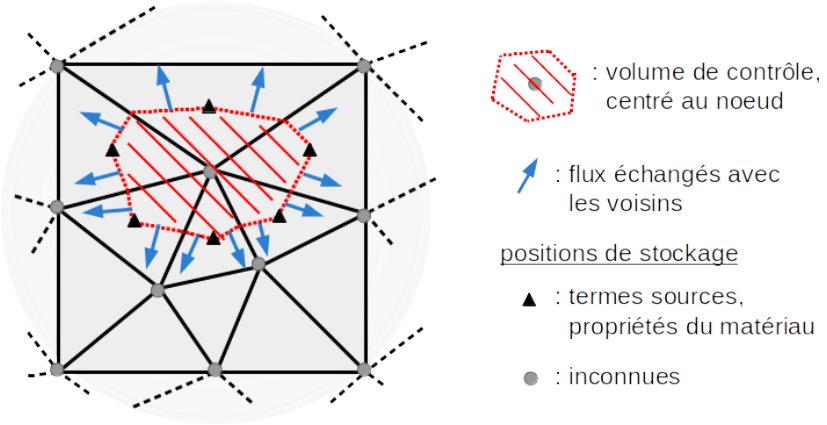


Figure 3.3 Volumes de contrôle et stockage pour le solveur de Helmholtz

Pour les trois physiques concernées, on considère une équation elliptique générale de la forme :

$$\bigtriangledown .\kappa\nabla\varphi + \beta\varphi = S, \tag{3.5}$$

où φ est la solution recherchée, κ un coefficient de diffusion et S un terme source. Certains termes deviennent nuls en fonction de la physique calculée. Pour obtenir la formulation

volumes finis, l'équation 3.5 est intégrée sur un volume de contrôle V centré au nœud, ce qui s'écrit :

$$\int_V \bigtriangledown.\kappa\nabla\varphi \; dV + \int_V \beta\varphi \; dV = \int_V S \; dV \tag{3.6}$$

En utilisant le théorème de la divergence, la première intégrale de l'équation 3.6 est transformée en intégrale de surface :

$$\oint_{\partial V} \kappa\nabla\varphi.\vec{\mathbf{n}} \; dS + \int_V \beta\varphi \; dV = \int_V S \; dV, \tag{3.7}$$

avec $\vec{\mathbf{n}}$ le vecteur unitaire normal sortant du volume de contrôle V.

Discrétisation

En utilisant le polynôme d'interpolation linéaire sur un triangle, on a :

$$\varphi = \alpha_1 + \alpha_2 z + \alpha_3 r, \tag{3.8}$$

avec r et z les directions radiales et axiales, respectivement. Les coefficients $\alpha_{1,2,3}$ peuvent être déterminés à partir des valeurs aux nœuds selon :

$$\alpha_1 = \frac{1}{2A}[a_i\varphi_i + a_j\varphi_j + a_k\varphi_k], \tag{3.9}$$

$$\alpha_2 = \frac{1}{2A}[b_i\varphi_i + b_j\varphi_j + b_k\varphi_k], \tag{3.10}$$

$$\alpha_3 = \frac{1}{2A}[c_i\varphi_i + c_j\varphi_j + c_k\varphi_k], \tag{3.11}$$

où A est l'aire du triangle défini par les trois sommets d'indices i, j, k. Les coefficients géométriques a_p, b_p et c_p (p = i, j, k) s'écrivent :

$$\left.\begin{array}{ll} a_i = & z_j r_k - z_k r_j \\ b_i = & r_j - r_k \\ c_i = & -(z_j - z_k) \end{array}\right\} \quad \text{avec un ordre de permutation naturel sur } i, j, k.$$

Finalement, on obtient la forme :

$$\varphi = N_i(r,z)\varphi_i + N_j(r,z)\varphi_j + N_k(r,z)\varphi_k, \tag{3.12}$$

où

$$N_p(r,z) = \frac{1}{2A}[a_p + b_p z + c_p r], \quad (p = i, j, k),$$

En prenant en considération les N nœuds voisins autour du nœud i, l'équation correspondante s'écrit :

$$A_i\varphi_i + \sum_{v=1}^{N} A_v\varphi_v = b_i. \tag{3.13}$$

L'assemblage du système algébrique d'équations donne alors :

$$\mathbf{A}\vec{\varphi} = \vec{b} \tag{3.14}$$

Le système linéaire 3.14 est résolu dans MC^3 par une méthode directe basée sur une décomposition LU. Les matrices creuses, symétriques et définies positives sont stockées selon le format ligne de ciel (*skyline*) qui s'adapte bien à la méthode LU sans pivot comme utilisée.

Les termes sources d'arc proviennent des différentes physiques résolues avec le solveur de Helmholtz introduit ci-dessus, excepté pour le calcul du rayonnement par la méthode FVM.

A. Chauffage ohmique

Par application de la loi d'Ohm avec l'équation de continuité stationnaire de distribution du courant, l'équation résolue pour le calcul du potentiel électrique ϕ durant la phase d'arc est :

$$\nabla \cdot \sigma\nabla\phi = 0, \tag{3.15}$$

où σ est la conductivité électrique du matériau. Cette équation est résolue dans la région du plasma où $\sigma \neq 0$ en considérant des conditions aux limites de type Dirichlet sur la surface des électrodes et de type Neumann nul sur les frontières de l'arc. L'intégration sur un volume de contrôle arbitraire V et l'application du théorème de la divergence donne la forme FVM :

$$\oint_{\partial V} \sigma\nabla\phi \cdot \vec{\mathbf{n}}\, dS = 0 \tag{3.16}$$

La résolution du potentiel électrique permet ensuite de calculer le champ électrique avec :

$$E = -\nabla\phi, \tag{3.17}$$

ce qui autorise finalement le calcul du terme source ohmique (effet Joule dû au passage du courant) :

$$S_{ohm} \;=\; \sigma|\vec{\mathbf{E}}|^2 \tag{3.18}$$

Le terme source d'arc S_{ohm} contribue à $\vec{S_{arc}}$ dans l'équation de conservation de l'énergie (3.1). La conductivité électrique du gaz dépend des valeurs locales du mélange d'espèces chimiques, de la température et de la pression, dont la valeur est interpolée dans une table de données fournie par un laboratoire spécialisé.

B. Force de Lorentz

Le champ magnétique auto-induit de l'arc est purement azimutal puisque la densité de courant électrique $\vec{\mathbf{J}}$ est contenue dans le plan (r,z). Les forces électromagnétiques sont donc aussi dans la plan (r,z) et provoquent une compression de l'arc (*Z-Pinch effect*). La fréquence du courant d'arc des disjoncteurs haute-tension est assimilable à la fréquence du réseau électrique (50Hz ou 60Hz généralement, 16.6Hz rarement). Les termes non-stationnaires des équations de Maxwell induisent donc des variations négligeables par rapport au cas stationnaire et peuvent donc être négligés. En présence d'un plasma neutre, les équations de Maxwell se réduisent donc à :

$$\nabla \cdot \vec{\mathbf{B}} = 0, \tag{3.19}$$

$$\nabla \cdot \vec{\mathbf{J}} = 0, \tag{3.20}$$

$$\nabla \times \vec{\mathbf{E}} = 0, \tag{3.21}$$

$$\nabla \times \vec{\mathbf{H}} = \vec{\mathbf{J}}. \tag{3.22}$$

Avec $\vec{H}$ le champ magnétique auto-induit et $\vec{B}$ la densité de flux magnétique. Le champ magnétique et la densité de flux magnétique sont liés par la relation (dans le vide) :

$$B_\theta = \mu_0 H_\theta, \tag{3.23}$$

où μ_0 est la perméabilité magnétique dans le vide. La perméabilité magnétique varie extrêmement peu en fonction des gaz, on utilise donc la perméabilité magnétique du vide pour MC^3.

En écrivant le rotationnel de la quatrième équation de Maxwell (3.22) on a :

$$\nabla(\nabla \cdot \vec{\mathbf{H}}) - \nabla^2 \vec{\mathbf{H}} = \nabla \times \vec{\mathbf{J}}, \tag{3.24}$$

ce qui se simplifie avec l'hypothèse d'un champ purement azimutal (la divergence du champ est nulle $\nabla.\vec{\mathbf{H}} = \nabla.\vec{\mathbf{B}} = \vec{0}$) en :

$$-\nabla^2 \vec{\mathbf{H}} = \nabla \times \vec{\mathbf{J}}. \tag{3.25}$$

Dans un repère cylindrique, la projection de cette équation vectorielle dans les directions axiales et radiales donne un résultat nul, tandis que dans la direction azimutale cela donne :

$$-\frac{1}{r}\frac{\partial H}{\partial r}-\frac{\partial^2 H_0}{\partial^2 r}-\frac{\partial^2 H_0}{\partial^2 z}+\frac{H_0}{r^2}=\nabla.(\vec{\mathbf{J}}\times\vec{\theta})+\frac{J_z}{r} \tag{3.26}$$

Avec

$$\frac{1}{r}\frac{\partial H}{\partial r}+\frac{\partial^2 H_0}{\partial^2 r}+\frac{\partial^2 H_0}{\partial^2 z}=\nabla.\nabla H_0, \tag{3.27}$$

l'écriture plus compacte de l'équation 3.26 est :

$$-\nabla.\nabla H_0+\frac{H_0}{r^2}=\nabla.(\vec{\mathbf{J}}\times\vec{\theta})+\frac{J_z}{r} \tag{3.28}$$

Après l'intégration sur un volume de contrôle arbitraire V, l'application du théorème de la divergence et le replacement de $\vec{\mathbf{J}}=\sigma\vec{\mathbf{E}}$, la forme intégrale (FVM) de cette équation est :

$$\oint_{\partial V}\nabla H_\theta\cdot\vec{\mathbf{n}}\;dS+\int_V\frac{H_\theta}{r^2}\;dV \;=\; \oint_{\partial V}(\sigma\vec{\mathbf{E}}\times\vec{\theta})\cdot\vec{\mathbf{n}}\;dS+\int_V\sigma\vec{\mathbf{E}}\cdot\frac{\vec{\mathbf{z}}}{r}\;dV. \tag{3.29}$$

Le solveur de Helmholtz décrit à la section précédente est une nouvelle fois utilisé pour résoudre cette équation. Des conditions aux limites de type homogène sont utilisées en conjonction avec des conditions de type Dirichlet sur l'axe de symétrie ($H=0$). Lorsque la distribution de H_0 est résolue, B_0 est calculé selon (3.23) puis les termes sources d'arc (forces électromagnétiques) sont obtenus par :

$$S_z = J_r B_\theta \tag{3.30}$$

$$S_r = -J_z B_\theta \tag{3.31}$$

Ils contribuent aux termes sources d'arc $\vec{S_{arc}}$ des équations de conservation de la quantité de mouvement (3.1).

C. Rayonnement thermique : modèle P1

Parmi les phénomènes énergétiques qui surviennent lors de l'opération d'ouverture d'un disjoncteur, le rayonnement thermique demeure celui qui prédomine à fort courant. Cependant, compte tenu des divers aspects qui entrent en jeu, c'est l'un des plus difficile à modéliser. En effet, pour modéliser précisément le phénomène de rayonnement, il faut prendre en considération sa nature spectrale et sa dépendance avec la température. De plus, l'arc électrique se

forme dans un gaz et du fait de l'élévation de température importante, de l'ordre de 25 000K, l'état de la matière considéré pour l'arc est alors le plasma thermique. Ce qui complique davantage la modélisation du rayonnement, car les propriétés radiatives peuvent varier de plusieurs ordres de grandeurs et le spectre optique est composé d'une partie continue sur laquelle se superposent de très nombreuses raies de résonance, tel qu'illustré à la figure 3.4.

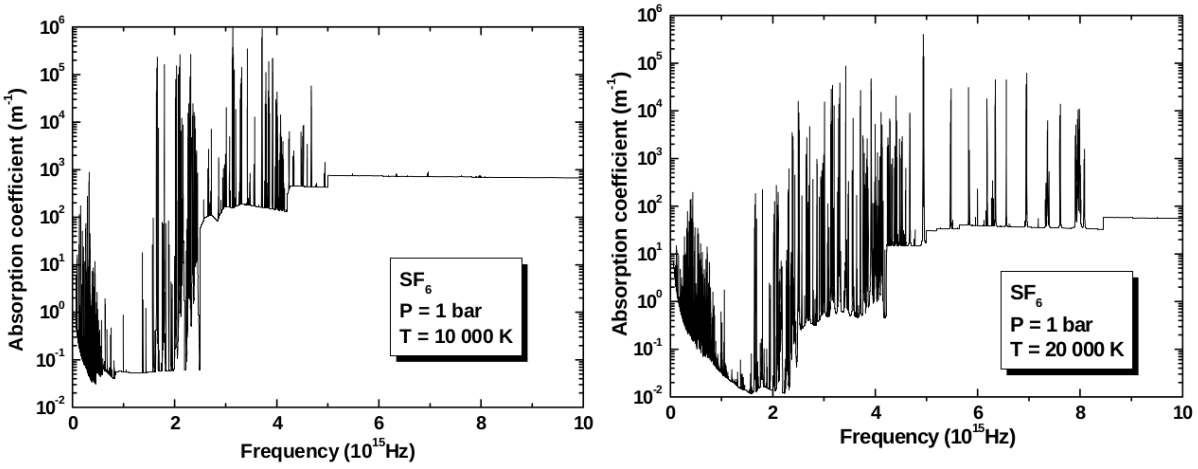


Figure 3.4 Coefficient d'absorption du SF6 à pression atmosphérique pour des températures de 10 000K et 20 000K - tiré de Randrianandraina [97]

Pour plus de détails et de notions sur le rayonnement thermique, on invite le lecteur à se rendre à l'annexe B.

Modéliser le rayonnement dans les arcs de disjoncteurs

Dans les disjoncteurs, le rayonnement se localise principalement dans les régions les plus chaudes du plasma ($T > 5000K$) où un échange intensif d'énergie par rayonnement a lieu. Il participe également à l'ablation de la buse (Figure 2.1). Les flux de chaleur aux parois ablatées de la buse doivent être évalués avec précision puisque l'ablation correspondante est principalement due au rayonnement reçu (Godin et al. [36], Martin et al. [71]). Dans les disjoncteurs basse tension, l'influence du phénomène d'ablation, est beaucoup moins significative, même parfois négligeable, du fait, d'une part des matériaux utilisés (principalement à base de polymères) et d'autre part de l'intensité du rayonnement beaucoup moins importante, indirectement liée au courant d'arc.

Concernant MC^3, les modèles P1 et FVM sont implémentés. Dans les prochains paragraphes, nous aborderons la description de la méthode P1 tandis que la méthode FVM sera décrite

dans la section 3.2.3.

La méthode P1, proposée initialement par Eby [25] pour l'application dans le cadre des disjoncteurs, est telle que son implémentation apporte une grande simplification de l'équation de transfert radiatif. En effet, l'hypothèse de rayonnement incident isotrope permet de remplacer l'équation intégro-différentielle de transfert radiatif (RTE) par une équation elliptique de type Helmholtz pour le rayonnement incident. Un autre avantage de la méthode P1 vient de son habilité naturelle à fournir une prédiction du rayonnement incident au niveau des parois. Cependant, comme il s'agit d'une équation de diffusion, le phénomène de transport à longue distance peut être mal prédit par la méthode P1. Par opposition, si le comportement du milieu participant est optiquement très épais alors le comportement propagatif tend plutôt à s'effacer devant un comportement plus diffusif, rendant le modèle P1 très attrayant pour ce genre d'application (d'un point de vue du rapport du coût de calcul et de la fidélité de prédiction).

Description des équations :

L'idée de base derrière la méthode P1 est d'exprimer l'intensité radiative comme des séries de Fourrier par la méthode de la séparation des variables. Ainsi on obtient une fonction dépendante de la localisation spatiale et une autre dépendante de la direction des rayons :

$$I_\nu(r,\hat{s}) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} I_l^m(r) Y_l^m(\hat{s}) \tag{3.32}$$

où $I_l^m(r)$ correspond aux coefficients de position et Y_l^m aux harmoniques sphériques. Ces séries définissent la base de la méthode des harmoniques sphériques, aussi notées P_N. Dans le cas du modèle P1, les séries sont tronquées à l'ordre $l = 1$. À travers un développement mathématique, Modest [79] montre que l'intensité radiative peut être calculée à partir du rayonnement incident G_ν et du flux radiatif q_ν, telle que :

$$I_\nu(r,\hat{s}) = \frac{1}{4\pi}(G_\nu + 3q_\nu \cdot \hat{s}) \tag{3.33}$$

où le rayonnement incident G_ν et le flux radiatif q_ν s'expriment tels que :

$$G_\nu = \int_{4\pi} I_\nu(r,\hat{s})\ d\Omega\ , \qquad q_\nu = \int_{4\pi} I_\nu(r,\hat{s})\hat{s}\ d\Omega \tag{3.34}$$

Avec les hypothèses faites au paragraphe précédent, en substituant Équ.(3.33) dans l'équation

de transfert radiatif (A.3) et en intégrant sur toutes les directions, on obtient les équations :

$$\nabla G_\nu = -3\kappa_\nu q_\nu \tag{3.35}$$

$$\nabla \cdot q_\nu = S_{rad} = \kappa_\nu(4\pi I_{b\nu} - G_\nu) \tag{3.36}$$

qui lorsque combinées, forment l'équation de type Helmholtz gouvernant le calcul du rayonnement incident G_ν et propre au modèle P1 :

$$\nabla \cdot (\frac{1}{\kappa_\nu}\nabla G_\nu) = 3\kappa_\nu(G_\nu - 4\pi I_{b\nu}) \tag{3.37}$$

L'intégration sur un volume de contrôle arbitraire V donne la formulation :

$$\oint_{\partial V} \frac{1}{\kappa_\nu}\nabla G_\nu \cdot \vec{\mathbf{n}}\ dS - \int_V 3\kappa_\nu G_\nu\ dV = \int_V -12\pi\kappa I_{b\nu}\ dV \tag{3.38}$$

Le solveur générique de type Helmholtz décrit plus haut est de nouveau employé pour la résolution du rayonnement par la méthode P1 dans MC^3 (pour chacune des bandes d'absorption).

Conditions aux limites :

Une des principales difficultés pour les applications pratiques de ce modèle est le manque de conditions aux limites réalistes vis-à-vis de la physique. Pour une condition d'axi-symétrie on impose généralement un flux nul sur l'axe de symétrie. Pour les parois de la buse, la condition aux limites la plus appropriée est la condition de Marshak :

$$\frac{1}{\kappa}\nabla G_\nu \cdot \hat{n} = -\frac{3}{2}\frac{\epsilon_\omega}{2-\epsilon_\omega}G \tag{3.39}$$

La condition de Marshak rajoute de la diffusion supplémentaire le long de la paroi correspondante et de l'absorption à travers. ϵ_ω est souvent pris égal à 1, les parois sont alors traitées comme étant un corps noir et non réflectives.

Discrétisation :

Dans le cadre de la simulation de l'opération des disjoncteurs, l'équation (3.37) est généralement résolue par une discrétisation par volumes finis sur des maillages de triangles comme c'est le cas dans MC^3. Les volumes de contrôles sont basés sur les nœuds où le rayonnement incident G est résolu. Ainsi, le rayonnement incident varie linéairement sur les triangles et est continu entre les triangles, tandis que les valeurs du coefficient d'absorption et l'émission de corps noir sont constantes sur les triangles. Afin de réduire la quantité de calculs à réaliser, l'intégration spectrale n'est pas faite sur une infinité de fréquences. C'est à dire que les coefficients d'absorption et d'émission sont approximés à l'aide de coefficients moyens par bandes de fréquences, généralement 5 à 9 bandes pour le SF6 et de l'ordre de 9 à 12 bandes pour le CO2 (pour plus de détails sur ce thème, voir les travaux de Gleizes, très riches sur le sujet et notamment [97, 49]). En suivant les développements de Eby [25], la résolution se ramène à un système matriciel creux dont les inconnues sont les valeurs du rayonnement incident G aux nœuds.

Le modèle P1 est certainement le modèle qui est le plus populaire dans la communauté des disjoncteurs et reste une référence indéniable dans le cadre du calcul du rayonnement dans les disjoncteurs, pour ne citer que quelques auteurs parmi les nombreux qui l'utilisent : [39, 67, 68, 87, 37, 80, 48]. Il est très utilisé aussi bien dans le cadre des disjoncteurs à basse-tension qu'à haute-tension.

D. Ablation

L'ablation dans les designs modernes de disjoncteurs haute-tension est une stratégie délibérée pour augmenter la pression dans le volume d'expansion thermique et ainsi faciliter l'extinction de l'arc. La qualité de la prédiction de l'ablation est donc cruciale. La sublimation de la buse, composée de Téflon, est la principale source de génération de gaz additionnel qui est principalement due au rayonnement incident à la paroi. La sublimation des parties métalliques à base d'alliages d'aluminium et des électrodes en tungstène et en cuivre produit l'autre partie. Dans MC^3, on suppose que l'ablation est causée uniquement par le flux thermique de rayonnement sur la buse, qui est obtenu soit par l'approximation P1, soit par l'approximation FVM (voir section suivante). Le changement d'état de la buse est lié à son enthalpie de vaporisation h_v.

Le débit de masse généré est :

$$\dot{m}_k = \left\{ \frac{q_{nk}}{h_v + e_f - e_i}, 0 \right\}, \tag{3.40}$$

avec q_{nk} le flux de rayonnement incident normal au side k,
h_v l'enthalpie de sublimation du Téflon,
e_i l'énergie interne du gaz à 1500K après sublimation,
e_f l'énergie interne finale correspondante à 3500K ([109]), la température finale des gaz. Le terme source à cumuler dans $\vec{S_{arc}}$ pour l'équation de conservation de la masse du système (3.1) est :

$$S_m = \frac{1}{\Delta V} \sum_{k=1}^{N_{sides}} \dot{m}_k L_k \tag{3.41}$$

Et, le terme source dans l'équation de conservation de l'énergie (3.1) est calculé selon :

$$S_{abl} = S_m \times e_f \tag{3.42}$$

3.2.3 Rayonnement thermique : modèle FVM-DOM

Contrairement au modèle P1 qui résout une approximation de l'équation de transfert radiatif (RTE), la méthode FVM (*Finite Volume Method*) se base directement sur une discrétisation de celle-ci. Cette méthode est candidate pour le remplacement de la méthode P1 pour les simulations disjoncteurs. Initialement proposée par Briggs et al. [16] dans le cadre de la simulation du transport de neutrons, cette approche a ensuite été adoptée par des pionniers tels que Raithby et Chui [96] et Chai et al. [19]. L'idée principale n'est pas de discrétiser seulement le domaine spatial, mais également celui des directions.
La première étape est d'adapter le schéma du 3D vers l'axi-symétrique, permettant ainsi de réduire le nombre d'inconnues. En effet, on considère que la solution CFD est axi-symétrique, ce qui est le cas pour tous les travaux dans le domaine de la modélisation des disjoncteurs à haute-tension. Ainsi, le champ de température peut être décrit comme une fonction $f(r, z)$, indépendante de la coordonnée azimutale. Cette hypothèse de symétrie ne peut pas être directement appliquée au contexte du rayonnement qui dépend de multiples variables. En effet, de par la nature fondamentale du rayonnement qui se propage le long de lignes droites dans tout l'espace physique, l'équation de transfert radiatif (RTE) ne peut pas être confinée à un plan de symétrie à la manière de la CFD. Comme illustré à la figure 3.5, la relation entre le champ de rayonnement entre deux points A et B, en considérant la condition d'axi-symétrie, induit seulement une rotation de la discrétisation directionnelle locale dans le plan (r, Φ), d'un angle $\Delta\Phi$. Même si les champs d'intensité radiative I_A et I_B ont des valeurs absolues différentes, il existe une relation qui peut être exploitée pour développer une approche axi-symétrique dans la formulation du problème de rayonnement.

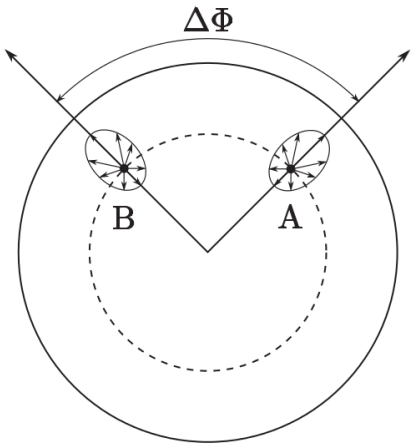


Figure 3.5 Condition axi-symétrique pour le modèle FVM - tiré de Melot et al. [76]

En appliquant les mêmes hypothèses que pour le modèle P1 (décrit aux paragraphes précédents), l'équation simplifiée s'écrit :

$$\nabla I_\nu|_{\hat{s}} = \kappa_\nu(I_{b\nu} - I_\nu) \tag{3.43}$$

Un volume de contrôle sur maillage non-structuré en 2D est balayé selon un angle $\Delta\Phi$, comme montré à la figure 3.6, et les angles solides de contrôle $N_\theta \times N_\psi$ sont équi-répartis par pas $\Delta\theta$ et $\Delta\psi$, respectivement, tels qu'illustrés à la figure 3.7.

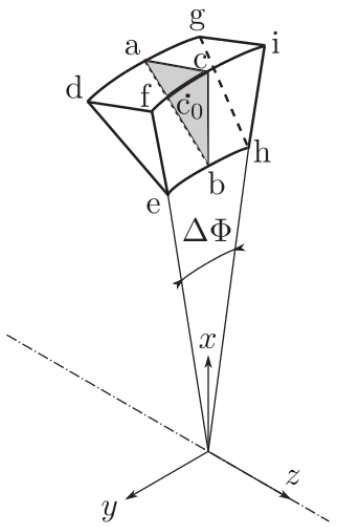


Figure 3.6 Volume de contrôle pour le modèle FVM - tiré de Melot et al. [76]

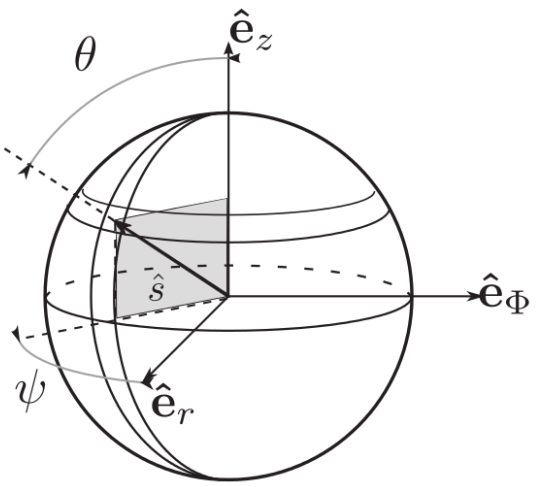


Figure 3.7 Angle solide de contrôle pour le modèle FVM - tiré de Melot et al. [76]

En intégrant l'équation 3.43 sur le volume de contrôle et en appliquant le théorème de Gauss, on obtient le schéma discret :

$$\sum_{faces} J_f^{l,m} I_f^{l,m} = (\kappa I_{bo}^{l,m} - \kappa I_o^{l,m}) \omega^{l,m} \Delta V_o \tag{3.44}$$

avec :

* l'indice f correspondant aux faces du volume de contrôle 3D,
* les indices l et m à la paramétrisation de la direction $\hat{s}(\theta_l, \psi_m)$,
* $I_f^{l,m}$ l'intensité radiative aux centres des faces du volume de contrôle
* $I_o^{l,m}$ l'intensité radiative au centre du volume de contrôle,
* $I_{bo}^{l,m}$ l'intensité radiative du corps noir, qui dépend de la température, au centre du volume de contrôle,
* ΔV_o le volume physique du volume de contrôle.

L'angle solide $\omega^{l,m}$ est donné par :

$$\omega^{l,m} = \int_{\Delta\psi} \int_{\Delta\theta} sin\theta \; d\theta \; d\psi = 2 \; sin\theta_i \; sin(\frac{\Delta\theta}{2}) \Delta\psi \tag{3.45}$$

Finalement, $J_f^{l,m}$ est le facteur géométrique défini par :

$$J_f^{l,m} = \mathbf{A} \cdot \int_{\Delta\psi} \int_{\Delta\theta} \hat{s}(\theta_l, \psi_m) \; sin\theta \; d\theta \; d\psi = \mathbf{A} \cdot \mathbf{S}^{l,m} \tag{3.46}$$

où $\mathbf{A} = A.\hat{n}$ et avec les coordonnées du vecteur de direction (illustrées à la figure 3.8) telles que :

$$\hat{s}(\theta_l, \psi_m) = sin\theta_l \; cos\psi_m \; \hat{e}_r + sin\theta_l \; sin\psi_m \; \hat{e}_\Phi + cos \; \theta_l \; \hat{e}_z \tag{3.47}$$

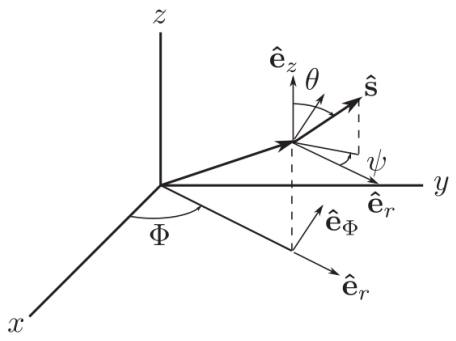


Figure 3.8 Repères pour les coordonnées de directions pour le modèle FVM - tiré de Melot et al. [76]

Les intensités aux faces sont dérivées selon un schéma amont classique, ce qui signifie que si une direction $\hat{s}$ entre dans le volume de contrôle alors $I_f^{l,m} = I_{amont}^{l,m}$, ou $I_f^{l,m} = I_o$ sinon. Avec un mapping considérant $\Delta\Phi = \Delta\psi$, tel qu'indiqué à la figure 3.9, il est alors possible de calculer les intensités radiatives à partir des quantités dans le plan $\Phi = 0$. Ainsi, l'hypothèse d'axi-symétrie, pour toutes les directions s'écrit :

$$I^{l,m}|_{\Phi=0} = I^{l,m}|_{\Phi=\Delta\Phi} = I^{l,m}|_{\Phi=-\Delta\Phi} \tag{3.48}$$

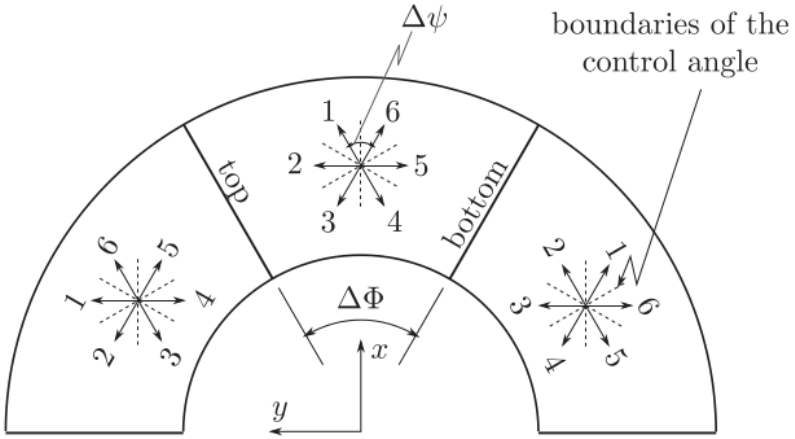


Figure 3.9 Mapping entre la discrétisation directionnelle et la discrétisation spatiale pour le modèle FVM - tiré de Melot et al. [76]

A cause de la condition de symétrie du plan $(\mathbf{e}_x, \mathbf{e}_y)$, pour le volume de contrôle central, les relations suivantes s'appliquent :

$$I^{1,m}|_{\Phi=0} = I^{6,m}|_{\Phi=0}, \quad I^{2,m}|_{\Phi=0} = I^{5,m}|_{\Phi=0} \;\; et \;\; I^{3,m}|_{\Phi=0} = I^{4,m}|_{\Phi=0} \tag{3.49}$$

Enfin, les intensités radiatives en-dessus et en-dessous des faces (**dfe**) et (**ghi**), comme indiquées à la figure 3.6, peuvent aussi être exprimées dans le plan $(\mathbf{e}_x, \mathbf{e}_y)$ en remarquant que pour les directions $l = 3$, pour le volume de contrôle central, la moitié des directions couvertes par l'angle de contrôle sont entrantes alors que l'autre moitié est sortante. Ainsi certaines directions ne participent pas au bilan d'énergie :

$$I_{haut}^{l,m} = I_o^{l,m}, \quad \forall l \; 2 \leq \; m \; \leq N_\psi \tag{3.50}$$

$$I_{bas}^{l,m} = I_o^{l,m+1}, \quad \forall l \; 1 \leq \; m \; \leq N_\psi - 1 \tag{3.51}$$

$$I_{bas}^{1,N_\psi} = I_o^{l,1} = 0, \quad \forall l \tag{3.52}$$

Ainsi, le schéma axi-symétrique pour le modèle FVM s'écrit :

$$(\sum_{faces\ lat.} J_f^{l,m} I_f^{l,m}) + J_{haut}^{l,m} I_{haut}^{l,m} + J_{bas}^{l,m} I_{bas}^{l,m} = (\kappa I_{bo}^{l,m} - \kappa I_o^{l,m}) \omega^{l,m} \Delta V_o \tag{3.53}$$

où, f dénote la relation aux faces (**dfig**,**fihe**,**dghe**) qui correspondent aux faces du maillage dans la discrétisation spatiale suivant le plan (r,z). Les faces supérieure et inférieure correspondent respectivement aux faces (**dfe**,**ghi**) de la figure 3.6. L'équation 3.53 doit être divisée par $\Delta\Phi$ pour être consistante avec la conservation de l'énergie. Finalement, en appliquant la condition d'axi-symétrie, cela nous permet d'éliminer la coordonnée Φ de la discrétisation spatiale et simplifie ainsi le problème initial composé de cinq variables par un problème à quatre variables. De plus, le plan de symétrie (r,z) permet de diminuer par moitié le nombre de directions pour les intensités radiatives. Au final, les simplifications découlant de l'axi-symétrie réduisent donc le nombre d'inconnues d'un facteur 2 $\times N_\theta$, où $N_\theta \times N_\psi$ correspond au nombre actuel de directions réparties sur 2π stéradians. Le coût de ces méthodes est très élevé même en axi-symétrie. En trois dimensions, le coût de ces méthodes est même trop important pour des simulations 3D réalistes (vis-à-vis de la complexité des géométries 3D) pour être praticables, et selon Christen [22], d'autres modèles devront être développés. En comparaison de la méthode P1 présentée précédemment, c'est la méthode la plus coûteuse en terme de temps de calcul, dépendamment du nombre de directions exigées de 10 x 10 directions minimum [76, 80, 48] pour être viable pour les simulations de disjoncteurs.

Le modèle de rayonnement FVM n'est pas employé actuellement lors des simulations réalisées par l'industriel à cause du seul fait que sa résolution est trop lente (de l'ordre de 20 sec par itération et par bande de coefficient moyen d'absorption du gaz). Cependant, on pense pouvoir l'accélérer suffisamment pour le rendre praticable, dans le prochain chapitre on propose quelques points d'amélioration pour atteindre cet objectif.

CHAPITRE 4 ANALYSE ET ACCÉLÉRATION DES MODÈLES

Au cours des années précédant cette thèse, durant une période d'un an, une entreprise composée principalement d'une équipe de spécialistes en informatique a été commissionnée pour paralléliser et accélérer le code MC^3. À l'issu de la prestation, l'accélération maximale proposée fut de 1.23x en utilisant 6 cœurs physiques de CPU. Ce facteur d'accélération représentait la limite de scalabilité, au-delà de ce nombre de cœurs, le code n'était plus accéléré mais ralenti. Finalement, cette prestation a donc été un échec.

Cette expérience démontre qu'une analyse plus complète et approfondie s'avère nécessaire et que la tâche d'accélération du code doit être confiée à des spécialistes en méthodes numériques, capables d'identifier les problématiques numériques associées aux modélisations et à leurs simulations, mais qui doivent être également à même d'en comprendre la physique modélisée afin d'en tirer bénéfice pour les tâches d'accélération. Une connaissance avancée des architectures de calcul haute performance, des modèles de programmation et des outils de profilage est également essentielle. Lorsqu'une composante est manquante parmi les trois : méthodes numériques ; physique ; informatique et architectures matérielles, toutes ou la plupart des opportunités d'accélération s'effacent.

Dans ce chapitre, une analyse approfondie est donc réalisée à travers ces trois composantes et des propositions d'améliorations algorithmiques sont introduites et analysées. Ce chapitre s'organise comme suit : une première section présente les notions relatives aux différents aspects de la modernisation de code d'un point de vue des modèles de programmation ; puis une deuxième section apporte quelques notions informatiques nécessaires au support des analyses, détaillées dans la troisième section.

4.1 Introduction sur la modernisation de code

Les ordinateurs haute performance modernes, qu'ils soient à l'échelle d'une grappe de calcul ou d'une station de travail, sont constitués d'une combinaison de diverses ressources. Ceci comprend : des processeurs multi-cœurs (Intel core i3/5/7, Intel Xeon, AMD Naples...), des accélérateurs de calculs (Xeon Phi Gen1, GPU, FPGA...), de larges caches, des unités logiques et arithmétiques vectorielles, des mémoires haute vitesse, des liens de communication inter-processeurs haut débit, etc. On assiste actuellement à une transition vers ces systèmes hybrides.
Ainsi, pour qu'il soit possible d'en tirer tous les avantages, la conception des logiciels haute

performance doit être réalisée avec soin. Des réglages pointus, des modifications architecturales, ou la création d'une nouvelle architecture pour l'application autorisent un maximum de performance sur les machines existantes ou futures. L'interdépendance entre l'usage efficient de ces ressources et les modèles de programmation est étroite ; le code source est donc critique pour tirer un maximum de performance d'une plateforme de calcul.

D'autre part, pour obtenir des résultats de calcul plus rapidement, de réaliser plus de calculs dans le même laps de temps ou de résoudre des problèmes de plus grande envergure, la conception d'une version parallèle d'une application devient indispensable. Du facteur d'accélération obtenu par rapport à la version séquentielle découle la mesure du succès de cette démarche. Afin de maximiser les performances, la part séquentielle du code doit être minimisée car elle ne sera pas accélérée avec l'augmentation du nombre de cœurs, les performances atteignent alors plus rapidement leurs limites quand cette part augmente.

Une conception moderne, efficace et haute performance du code doit prendre en considération tous les niveaux de parallélisme. C'est à dire prendre en compte le **parallélisme d'instructions** (**ILP**) (les unités d'exécution sur les entiers, les logiques et les nombres en virgule flottante sont indépendants), le **parallélisme vectoriel** (**VLP**) au niveau du cœur de processeur par l'utilisation effective des unités vectorielles (SIMD : Single Instruction Multiple Data, les mêmes instructions de calcul sont répétées pour des blocs de données différents), le **parallélisme de tâches** (**TLP**) où des threads au sein du cœur de processeur et/ou des multiples cœurs du même processeur coopèrent par équipe pour la réalisation d'une tâche d'un processus donné (parallélisme de type mémoire partagée, entre les threads) et enfin, le **parallélisme à mémoire distribuée** où des processus indépendants coopèrent et peuvent communiquer par l'envoi de messages.

Cependant, les CPUs, les GPUs et les accélérateurs (Xeon Phi Gen2, FPGA) possèdent des architectures de mémoire différentes qui doivent absolument être prises en considération dans la conception architecturale de l'application, dans les choix des algorithmes, dans les choix des structures de données, etc. En effet, les vitesses des unités de traitement diffèrent de plusieurs ordres de grandeur par rapport au temps d'accès aux données nécessaires aux calculs à charger de la mémoire vers les registres. Les unités de traitement réalisent les calculs uniquement entre les données stockées dans les registres. Les registres sont très peu nombreux ; par exemple, les processeurs de l'architecture Intel Kaby Lake, dispose de 168 registres pour les entiers et 172 registres pour les nombres en virgule flottante. Les accès à la mémoire sont donc fréquents que cela soit pour sauvegarder les valeurs stockées dans les registres vers la mémoire ou pour charger dans les registres des valeurs en mémoire. Pour minimiser la latence, c'est à dire le temps d'attente entre le moment où la donnée requise est effectivement

disponible et le moment où elle a été demandée par le cœur du processeur, des caches sont interposés entre les registres et la mémoire. Plus la latence est élevée et plus le processeur est en attente, sans réaliser de calculs. Plusieurs niveaux de caches sont disponibles et possèdent des latences et des débits différents. Par exemple, pour l'architecture Intel Kaby Lake [44] la mémoire s'organise telle qu'identifie le tableau 4.1. Comme on peut le voir dans le tableau

Tableau 4.1 Architecture mémoire typique d'un processeur X86_64

Niveau	Taille	Latence **minimum** [ns]	Débit (R/W/Copie) [Go/s]
registres (par cœur)	168int/172fp	0	n.a.
cache Level 1 (par cœur)	32Ko	≈ 1.3	$\approx 1400/750/1250$
cache Level 2 (par cœur)	256Ko	$\approx 6-8$	$\approx 450/250/50$
cache Level 3 (par processeur)	3-60Mo	$\approx 17-18$	$\approx 200/150/200$
mémoire principale 4 channels (DDR4-2400-15-15-15-35)	4-3000Go	≈ 70	$\approx 53/44/46$

4.1, plus on se rapproche des registres en descendant dans les niveaux et plus la latence est faible avec des débits en forte augmentation. Cependant, les tailles de stockage sont de plus en plus petites. Les caches sont organisés par lignes de 64 bytes, c'est à dire qu'une ligne peut contenir 16 entiers ou nombres en virgule flottante simple précision ou encore 8 en double précision. Lorsque les données en mémoire ne sont pas accédées de manière contiguë, sont non-alignées avec ces lignes ou se trouvent dans des lignes parsemées du cache, l'impact sur les performances est extrêmement négatif. Les algorithmes et leur(s) implémentation(s) doivent donc prendre en considération ces aspects pour maximiser la présence dans le cache des données requises aux calculs aux moments où elles sont nécessaires. Les structures de données doivent être organisées pour être *cache friendly*, c'est à dire favoriser les accès séquentiels et contigus, ce qui minimise leur temps d'accès et maximise la vitesse d'exécution d'un facteur loin d'être négligeable.

Pourquoi pas les GPUs ?

Les GPUs de calculs (comme par exemple les cartes de type NVidia Tesla ou AMD Radeon W9xxx) sont intéressants dans le sens où la bande passante mémoire disponible est de l'ordre de 4 à 10 fois supérieure à celle des CPUs classiques, ce qui permet d'approcher ce facteur d'accélération lorsque l'application est *memory bounded* (comme c'est le cas dans la plupart des modélisations de la physique et théoriquement pour MC^3). Lorsque l'application n'est pas limitée par les transferts mémoire, elle est dite *compute intensive*. Dans ce cas, le nombre d'opérations en virgule flottante (FLOPS) est alors la partie limitant les performances. Les GPUs ou les Xeon Phi mettent à disposition $\approx 3\ TeraFLOPS$, tandis que les processeurs

disposant du plus grand nombre de cœurs atteignent $\approx 0.8\ TeraFLOPS$. La bande passante mémoire (débit) des GPUs est du même ordre de grandeur que la bande passante des accélérateurs (comme par exemple les Intel Xeon Phi). Cela est permis grâce à un niveau de mémoire supplémentaire, interposé entre la mémoire principale et les caches et dont la taille varie de 4 à 32 Go. Cependant, les Xeon Phi représentent une meilleure opportunité, puisqu'à performance mémoire équivalente des GPUs, ils permettent de minimiser les différences architecturales de l'application et des algorithmes à mettre en place en comparaison avec les CPUs classiques. Ainsi, optimiser une application pour CPUs classiques, l'optimise également pour accélérateurs Xeon Phi tout en mettant à profit les aspects architecturaux de la mémoire.

Tous ces aspects doivent être analysés de façon précise, notamment par l'utilisation d'outils et de méthodes de profilage statiques (analyse du code source, sans l'exécuter) et dynamiques (avec des mesures effectuées lors de l'exécution de l'application). La prochaine section présente une introduction sur l'analyse de performance afin de donner les clefs nécessaires à la compréhension de l'analyse de MC^3 détaillée dans la section 4.3.

4.2 Introduction sur l'analyse de performance et sur l'optimisation

Les interrogations générales :

- accélérer MC^3 d'un facteur intéressant, uniquement en parallélisant le code existant est-il possible ?
- MC^3 est-il de type *memory bandwidth bounded* ? ou *memory latence bounded* ?
- Les algorithmes originellement séquentiels sont-ils efficaces une fois parallélisés ? ou faut-il utiliser des algorithmes parallèles plus efficaces ?
- Les structures de données sont-elles de type *cache friendly* ?
- Quel est le taux d'utilisation des unités vectorielles ?

sont autant de questions auxquelles il n'est pas trivial de répondre et l'utilisation d'outils et de méthodes de profilage est indispensable pour y répondre. Dans cette section, on ne se focalisera pas sur les outils utilisés eux-même mais plutôt sur les critères de décision auxquels s'intéresser et qui nous amèneront à la modernisation de MC^3. Cependant, la sélection de l'environnement de développement et d'optimisation est une décision qui aura une forte influence sur tout le processus d'optimisation. Un bon environnement fournira de bons outils de compilation, des bibliothèques optimisées et prêtes à l'usage, de bons outils de débogage et de profilage pour déterminer exactement ce que le code produit à l'exécution.

L'obtention d'une meilleure performance passe en partie par l'optimisation et la validation

du code source, **avant** l'ajout de la vectorisation et de la parallélisation. L'obtention d'un résultat juste doit être faite avec un nombre minimum d'opérations. Choisir la précision arithmétique des opérations en virgule flottante (simple vs double précision) qui s'avère nécessaire en fonction des données et des algorithmes, réduire l'impact des boucles (*unroll*, compteurs d'itérations constants dans la boucle), éviter ou minimiser les branchements conditionnels dans les algorithmes, éviter la répétition de calculs en utilisant les résultats calculés précédemment, éviter les sauts dans les algorithmes (*goto*), éviter ou minimiser l'utilisation de fonctions mathématiques coûteuses (logarithmes, exponentiels, racines, puissances) sont autant de points à considérer pour maximiser la probabilité d'obtenir un code performant.

Les principaux points d'analyse seront :

- **Efficacité d'utilisation du débit mémoire :**
  Le bus mémoire est-il saturé efficacement (atteinte du débit maximal) ?
- **Efficacité d'utilisation des caches :**
  Les requêtes de données du processeur sont-elles favorablement régulièrement satisfaites (données présentes dans le cache) ou doit-il régulièrement en attendre la réception ?
- **Équilibre de charge entre les threads :**
  Les threads réalisent chacun leur travail en parallèle. La durée d'exécution de chacun des threads est-elle équilibrée ou certains threads attentent-ils leurs camarades ?
- **Taux d'occupation des processeurs :**
  Le taux d'occupation effectif des processeurs est-il optimal ou les temps de communication et de synchronisation inter-threads sont-ils majoritaires ?
- **Taux de parallélisme :**
  La partie parallèle du code est-elle prédominante et en quantité optimale ?
- **Taux d'usage des unités vectorielles :**
  Les unités vectorielles permettent une parallélisation au niveau des cœurs eux-mêmes. Sont-elles utilisées ? Si oui, l'usage en est-il efficace ?
- **Efficacité des structures de données :**
  Les accès aux données sont-ils alignés en mémoire avec les lignes de caches ? Les lectures/écritures sont-elles contiguës ou aléatoires ?
- **Dépendance de données :**
  Les algorithmes comprennent-ils de longues chaînes de dépendance, des dépendances entre les différentes physiques ?
- **Pertinence du code source :**

Le code source maximise-il l'efficacité du compilateur pour la génération d'un exécutable performant ?

— **Pertinence des algorithmes :**
Les algorithmes exposent-ils au maximum le parallélisme et maximisent-ils la scalabilité ?

— **Taux de prédiction des branchements :**
Des unités dédiées au décodage des instructions cherchent à prédire à l'avance les bons choix de branches à suivre pour minimiser les latences d'accès aux données. Quels est le taux de bonnes prédictions ?

L'analyse de MC^3 dans la prochaine section permettra de répondre à ces questions.

4.3 Profilage et analyse CPU de MC^3

Les modèles pertinents ont été confirmés au chapitre 2 et sont pour la plupart déjà présents dans MC^3. De légères modifications, lorsque nécessaires, seront appliquées.
Les points généraux d'analyses évoqués dans la section précédente permettront de répondre notamment aux quelques questions générales : plus de threads seraient-ils utiles dans MC^3 ? Où faudrait-il ajouter du parallélisme ? Quels algorithmes sont à réviser ? Qu'est-ce qui va mal dans MC^3 ? Qu'est ce qui va bien ? Et finalement, pourrait-on accélérer MC^3 ou non ?

Dans une première section, nous procéderons à une analyse exploratoire : de courtes études ont été faites afin d'avoir un aperçu général du niveau d'optimisation, et donc aussi du niveau de vectorisation et parallélisation du solveur. Ensuite, un profilage plus précis a été réalisé sur la version séquentielle de MC^3, et les résultats correspondants sont présentés dans la section 4.3.2. Afin d'avoir une bonne vision d'ensemble du modèle global et de ses contraintes, les différentes équations et leur discrétisation ont été rappelées à la section 3.2, des propositions d'améliorations algorithmiques adaptées sont alors proposées.

Un cas test représentatif de la simulation de l'interruption d'un courant de court-circuit est réalisé durant un court instant. Celui-ci est suffisamment court pour limiter la durée d'exécution mais suffisamment long pour que les mesures de performances effectuées soient représentatives. Un temps d'exécution d'environ 30 à 90 secondes est tout à fait pertinent dans notre cas.

L'environnement de profilage et debugging sera basé sur une plateforme Linux. Il comporte l'utilisation des outils GNU (gdb, gprof, gfortran, g++), PGI et Intel Parallel Studio (ifort, icpc, Vtune Profiler). Notre analyse se basera sur un code compilé avec le compilateur Intel. Comme les opérations de debuging et d'optimisation sont devenues indispensables au déve-

loppement d'application, le matériel informatique (CPUs, accélérateurs) incluent des unités dédiées aux mesures de performance et aux opérations de débogage. Les outils de type *PAPI* (**P**erformance **A**pplication **P**rogramming **I**nterface) ont été également employés pour obtenir les mesures réalisées par ces unités.

Cet environnement servira d'appui pour développer et justifier les modifications algorithmiques et d'implémentation proposées.

4.3.1 Analyse générale

Influence du nombre de threads :

Pour cette étude, la version parallèle et mono-espèces de MC^3 livrée par le prestataire chargé de l'accélérer a été utilisée. En faisant varier le nombre de threads et en comparant la durée d'exécution ou le facteur d'accélération, on peut rapidement connaître la limite de scalabilté de MC^3. Le tableau 4.2 et la figure 4.1 résument ces résultats :

Tableau 4.2 Test de scalabilité de MC^3

Nombre de threads	Accélération	Efficacité parallèle
2	1.15x	57.5 %
4	1.20x	30.0 %
6	**1.23x**	20.5 %
8	1.02x	12.75 %
16	0.81x	−5.06 %

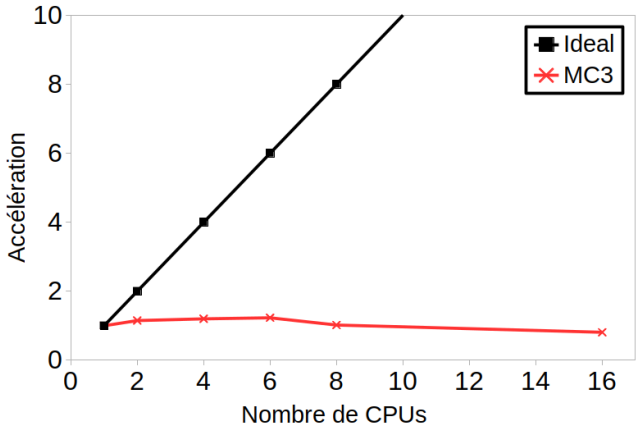


Figure 4.1 Accélération de MC^3 en fonction du nombre de CPUs

L'accélération et l'efficacité parallèle sont anormalement faibles. Le problème peut être lié à de multiples causes que nous allons tenter d'identifier. La charge de travail acheminée à un

des threads (ou à tous) est peut-être trop faible ou alors les threads passent trop de temps à se synchroniser plutôt qu'à travailler efficacement sur les tâches. Cela peut également être la cause d'une limitation due à la saturation de la bande passante mémoire, ajouter plus de threads n'apportant alors aucun gain. La taille du problème résolu ne peut pas être en cause dans notre cas, la charge de travail totale est assez importante pour que l'on s'attende à ce que l'utilisation de 32 threads (en mémoire partagée) soit potentiellement bénéfique.

Influence de la vectorisation :

Pour connaître l'impact sur les performances de l'utilisation des unités vectorielles, le code source a été compilé avec l'option *-no-simd* du compilateur Intel. Ce qui permet d'indiquer au compilateur de ne pas générer d'instructions processeur vectorielles. Ensuite, la comparaison des vitesses d'exécution AVEC vectorisation (activée par défaut à partir des options *-O2*) et SANS vectorisation, permet de connaître l'usage effectif des unités vectorielles. Les résultats révèlent une différence de vitesse d'exécution de l'ordre de seulement $\approx \mathbf{4-5}$ %. Cela peut signifier que les heuristiques du compilateur ne sont pas capables d'évaluer si les vectorisations des boucles sont sûres et donc il ne les vectorise pas, ou alors que l'efficacité de la vectorisation est très basse. Soit à cause de la complexité des algorithmes et du code source, soit en raison des chemins d'accès aux données en mémoire trop compliqués à prédire statiquement (la vectorisation est un parallélisme de données, non pas de tâches). La saturation de la bande passante mémoire peut aussi être la cause des faibles performances de la vectorisation dont nous vérifierons le taux réel d'usage par une analyse détaillée dans la section (4.4).

Complexité algorithmique :

Une manière de déterminer l'efficacité des algorithmes employés dans MC^3 est de mesurer les temps d'exécution en fonction de la taille des problèmes résolus. Cela permet d'évaluer la complexité algorithmique globale du code. Par exemple, si le nombre de cellules du maillage est doublé et que la complexité est de l'ordre de $O(n^2)$, alors le temps d'exécution sera multiplié par quatre, si la complexité est plutôt de l'ordre de $O(n)$ alors le temps d'exécution sera seulement doublé et ainsi de suite. Cependant, les tailles des cellules du maillage influencent (la physique) le pas de temps et affectent donc le nombre d'itérations nécessaire pour atteindre le même temps final de simulation. En conséquence, dans le but de mesurer le coût réel par itération pour un problème d'une taille donnée sans être affecté par les phénomènes physiques, les mesures qui suivent sont réalisées à nombre d'itérations (pas de temps) constant et égal à 1000. Les résultats sont présentés aux figures 4.2 et 4.3.

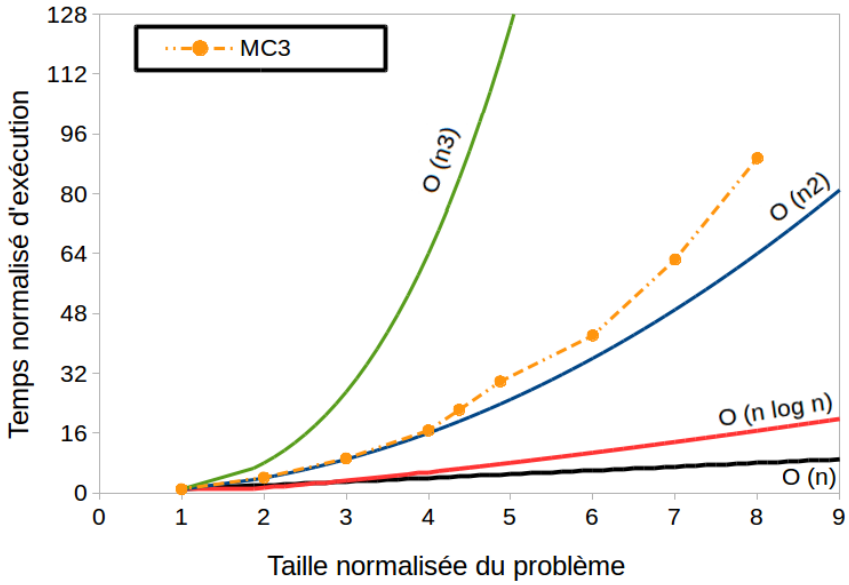


Figure 4.2 Complexité algorithmique de MC^3 en version séquentielle, AVEC adaptation de maillage et parois mobiles

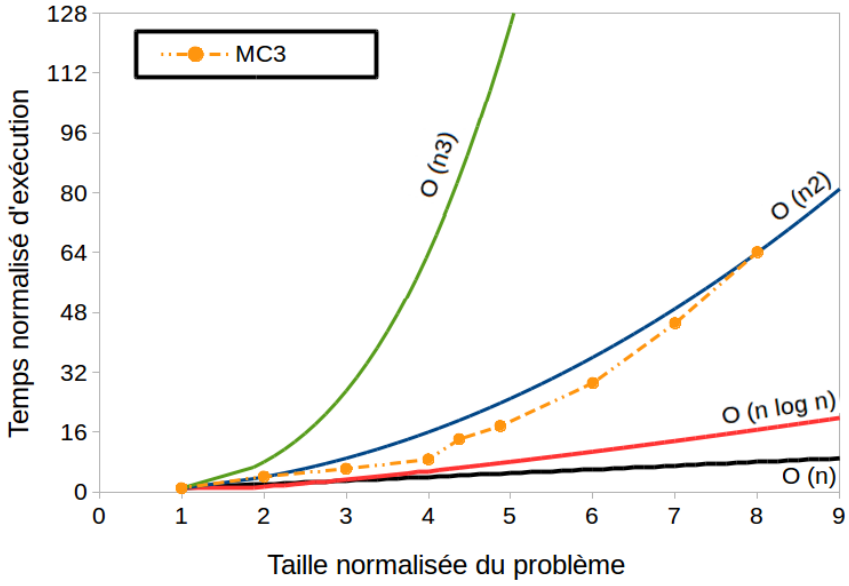


Figure 4.3 Complexité algorithmique de MC^3 en version séquentielle, SANS adaptation de maillage ni parois mobiles

La complexité algorithmique de MC^3 se situe approximativement à $\mathbf{O(n^2)}$, et est plus élevée encore lorsque l'adaptation de maillage et la gestion des parois mobiles sont activées. Cela nous laisse supposer que de meilleurs algorithmes sont envisageables pour améliorer les performances du solveur. L'idéal en pratique étant de se rapprocher au maximum d'une complexité linéaire. Il est généralement admis que des algorithmes d'une complexité inférieure ou égale à $O(n\ log\ n)$ sont de bons algorithmes d'un point vue de l'efficacité en temps de calcul.

Mesures globales :

Le profilage général de MC^3 permet de déterminer quelle est la physique la plus coûteuse d'un point de vue du temps de calcul et permet de connaître les proportions relatives de temps d'exécution de chacune des physiques. Les résultats sont présentés sur la figure 4.4 où l'on peut voir que la physique prépondérante est la mécanique des fluides.

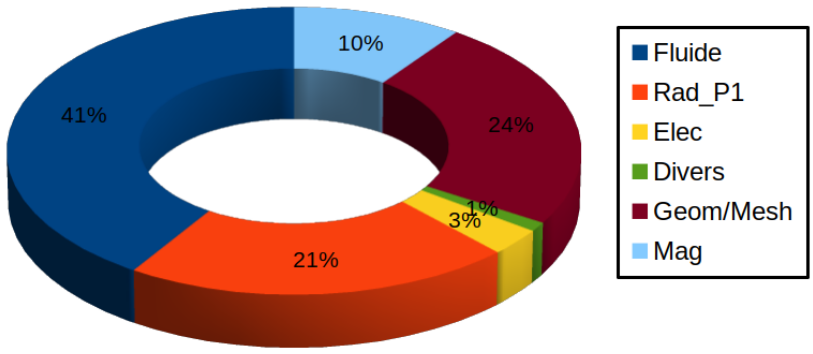


Figure 4.4 Profilage général de MC^3 en version séquentielle mono-espèces avec adaptation de maillage et parois mobiles

L'adaptation et la gestion de la mobilité du maillage représentent la deuxième part la plus importante du temps d'exécution, suivies de près par la résolution du rayonnement par la méthode P1. La partie électro-magnétique est la moins coûteuse.

Utilisation globale du CPU et du débit de la mémoire :

Les mesures de performances ont été réalisées sur un processeur Intel Core i7 6700k. Le processeur possède 4 cœurs physiques à une fréquence d'horloge de 4.0 GHz avec une cadence de jusqu'à 32 opérations SP (Simple Precision) par cycle et par cœur, ce qui correspond à un pic théorique d'opérations arithmétiques par seconde de 512 SP GFLOPS (256 DP GFLOPS (double précision)). L'utilisation actuelle est de 3.49 SP GFLOPS pour une valeur moyenne de 0.85 opérations SP par cycle (sur 32 opérations SP possibles). Le processeur est utilisé à moins de 0.07 % de ses capacités théoriques maximales et les unités arithmétiques sont utilisées à seulement 2.7 % de leurs capacités. La bande passante mémoire (Figure 4.5) disponible (32 GB/s sur notre système) est sous-utilisée. Elle a limité les performances en atteignant sa limite de débit moins de 0.8 % du temps total ($\approx 69\ s$).

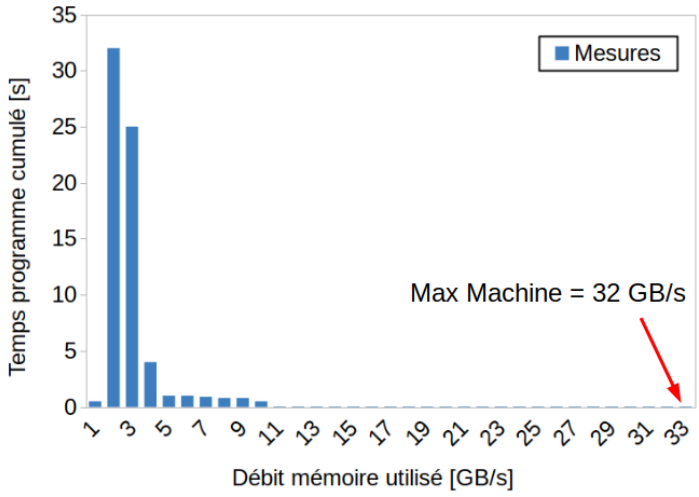


Figure 4.5 Temps passé en fonction de la bande passante mémoire

La majorité du temps, le débit mémoire maximal n'est pas atteint, car accéder à des portions de mémoire trop petites (et/ou aléatoires) en réalisant beaucoup d'accès ne permet pas d'atteindre le pic théorique maximal de débit mémoire. Le bus mémoire n'est alors pas saturé en débit mais en latence. Généralement, les limitations de performance d'une application sont liées : soit au débit d'instructions trop élevé ; soit à la latence mémoire ; soit à la saturation de la bande passante mémoire disponible ; soit au coût arithmétique. MC^3 souffre de ces quatre limitations (voir Figure 4.4), même simultanément dans certaines portions du programme. Le manque de vectorisation et de parallélisme limitent également les performances de l'application.

Taux CPI :

Le CPI (**C**ycles **P**er **I**nstruction retired), est une métrique de performance fondamentale, qui indique approximativement combien de temps (en cycles CPU) a pris chaque instruction exécutée. Comme les processeurs superscalaires modernes peuvent soumettre jusqu'à 4 instructions par cycle au scheduler, la valeur théorique idéale est de 0.25 cycle. Cependant, les chaines de dépendances, les effets de latence de la mémoire, les mauvaises prédictions de branches, etc, peuvent conduire à des valeurs de CPI sur-évaluées. En conséquence, une valeur de CPI $\leq$ 1.0, est admise comme acceptable pour les applications à haute-performance. Le CPI est très représentatif du potentiel d'accélération. Les fonctions de MC^3 nécessitant une attention particulière sont indiquées au tableau 4.3.

Tableau 4.3 Résumé des mesures de *CPI*

Fonctions	Catégories	Valeurs	Impacts
cfswno	maillage	8.0	fort
cfsvg0	maillage	5.8	faible
srcrad	rad	5.7	fort
calcxy	AX=Y	5.6	fort
cfsrsy	maillage	4.3	fort
calcrad	rad	3.3	fort
adusnb	maillage	3.3	médium
ptgauss	rad	3.2	fort
fction2	rad	2.5	fort
adyhdn	maillage	2.4	faible
intvol	rad	2.0	fort
adilct	maillage	1.8	faible
cfpcfl	fluide	1.5	médium
cfsisg	maillage	1.5	médium
fction1	rad	1.5	médium
ddot	AX=Y	1.4	fort
matcrs	AX=Y	1.2	fort
cfpmgd	maillage	1.2	fort
cfpbcd	fluide	1.1	médium

D'après ces résultats, toutes les physiques et la partie du maillage mobile adaptatif sont touchées. Les valeurs présentées révèlent beaucoup de potentiel d'accélération. En corrélation avec les autres métriques présentées en annexes, nous pourrons déterminer quelles sont les causes de ces CPI élevés. Dans tous les cas, cela indique que les algorithmes, les structures de données ou les accès en mémoire en place pour ces fonctions ne sont pas optimaux. En utilisant d'autres critères de mesures, d'autres fonctions critiques qui ne sont pas présentes dans cette liste apparaîtront, c'est pourquoi de nombreuses métriques différentes doivent être prises en compte pour l'analyse. Le résumé global des métriques est présenté dans la section suivante.

4.3.2 Profilage à l'aide des métriques CPU

Ces premières analyses globales révèlent un fort potentiel d'amélioration. Des mesures plus précises ont été réalisées en se basant sur des métriques CPU. Elles sont nécessaires pour déterminer quels sont les leviers prédominants sur lesquels nous allons agir pour libérer les performances des processeurs et exploiter au maximum leurs potentiels. Elles sont détaillées en annexe B.

Sur la figure 4.6 est représenté le pipeline basique des processeurs Intel de génération Skylake. Chacun des éléments matériels constituants le pipeline possèdent des métriques de performance. Ces métriques sont des événements matériels de profilage et de débogage, il peut s'agir de mesures temporelles, de compteurs d'événements quelconques, de capture d'exceptions, etc.

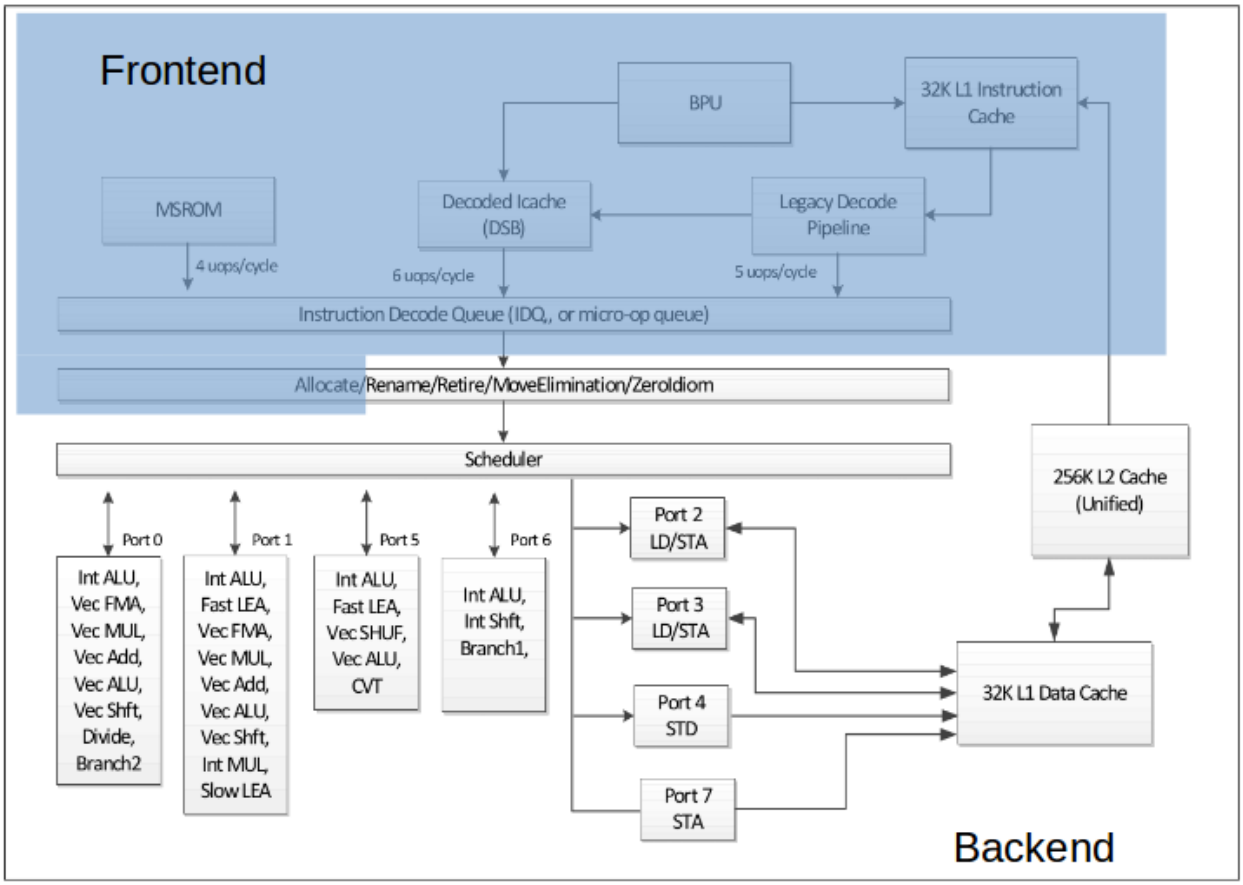
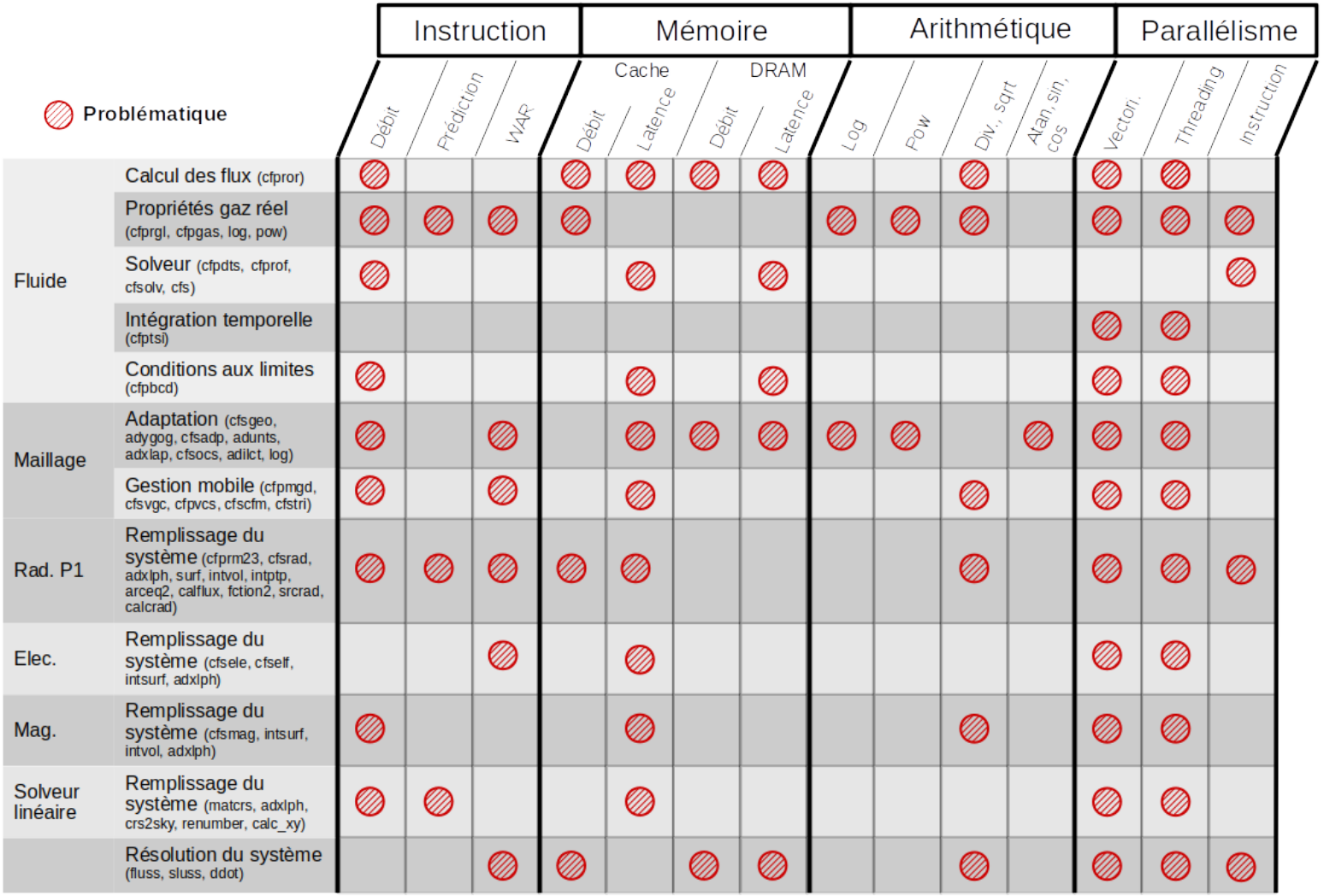


Figure 4.6 Pipeline du CPU Intel Skylake

De façon générale, un processeur saute d'une adresse mémoire à une autre où sont présentes des instructions. Les données et les instructions sont stockées indifféremment dans la mémoire vive. Pour minimiser les temps d'accès aux instructions suivantes, données et instructions sont copiées dans les caches. Le processeur possède des unités de décodage et de ré-ordonnancement des instructions afin, notamment, d'essayer de masquer la latence, d'éliminer les dépendances dans les chaines algorithmiques, pour globalement accélérer l'exécution. Ces unités font partie du front-end, représenté en bleu sur la figure 4.6. On peut interpréter cette partie comme la partie intelligente du processeur. Le reste des unités et des caches font partie du back-end et sont plutôt vus comme de simples exécutants, plus la partie mémoire (cache L2, DRAM...). Pour plus d'information sur le fonctionnement des processeurs, le lecteur peut consulter la documentation d'architecture Intel [44]. Front-end et back-end possèdent chacun des métriques qui révèlent les points faibles (et points forts) de MC^3 mesurés lors de l'exécution.

Le tableau 4.4 résume pour chaque physique les types de limitations encourues ; les catégories *instruction*, *mémoire* et *arithmétique* révèlent un excès tandis que la dernière catégorie *parallélisme* reflète d'un manque. Le lecteur intéressé par les détails de profilage des fonctions de MC^3 et par la liste des métriques utilisés peut se rendre en annexe B.

Tableau 4.4 Résumé des facteurs limitant les performances de MC^3



Finalement, les réponses aux principales interrogations sont symptomatiques :

— **Efficacité d'utilisation du débit mémoire :**
Le bus mémoire est-il saturé efficacement (atteinte du débit maximal) ? **NON**.

— **Efficacité d'utilisation des caches :**
Les requêtes de données du processeur sont-elles favorablement régulièrement satisfaites (données présentes dans le cache) ou doit-il régulièrement en attendre la réception ? **NON, LE PROCESSEUR ATTEND RÉGULIÈREMENT**.

— **Équilibre de charge entre les threads :**
UN THREAD !

— **Taux d'occupation des processeurs :**
Le taux d'occupation effectif des processeurs est-il optimal ou les temps de communication et de synchronisation inter-threads sont-ils majoritaires ? **UN THREAD !**

— **Taux de parallélisme :**
La partie parallèle du code est-elle prédominante et en quantité optimale ? **NON**.

— **Taux d'utilisation des unités vectorielles :**
Sont-elles utilisées ? **QUASIMENT PAS**. Si oui, l'usage en est-il efficace ? **NON**.

— **Efficacité des structures de données :**
Les accès aux données sont-ils alignés en mémoire avec les lignes de caches ? **NON**.
Les lectures/écritures sont-elles contiguës ou aléatoires ? **ALÉATOIRES**.

— **Dépendance de données :**
Les algorithmes comprennent-ils de longues chaînes de dépendance, des dépendances entre les différentes physiques ? **OUI**.

— **Pertinence du code source :**
Le code source maximise-il l'efficacité du compilateur pour la génération d'un exécutable performant ? **NON**.

— **Pertinence des algorithmes :**
Les algorithmes exposent-ils au maximum le parallélisme et maximisent-ils la scalabilité ? **NON**.

— **Taux de prédiction des branchements :**
Quels est le taux de bonnes prédictions ? **FAIBLE**.

MC^3 est de type : *memory latence bounded + memory bandwidth bounded + instruction bounded*, ce qui rend particulièrement délicate la tâche d'accélération. De plus, augmenter le nombre de threads ou paralléliser plus de boucles est donc inutile. Le programme doit être révisé dans son ensemble, de son architecture à son implémentation, en passant par ses algorithmes. En effet, comme on peut s'en rendre compte à la figure 4.7, les fonctions sont très entremêlées et non-structurées (ce qui explique en partie les mauvaises performances de MC^3). Suite à ces analyses et constatations, les points les plus coûteux en temps de calcul et donc les plus importants pour l'accélération du code et sur lesquels il faudra se concentrer en priorité sont connus. Dans la prochaine section, des propositions sont faites pour améliorer la rapidité d'exécution de ces points chauds.

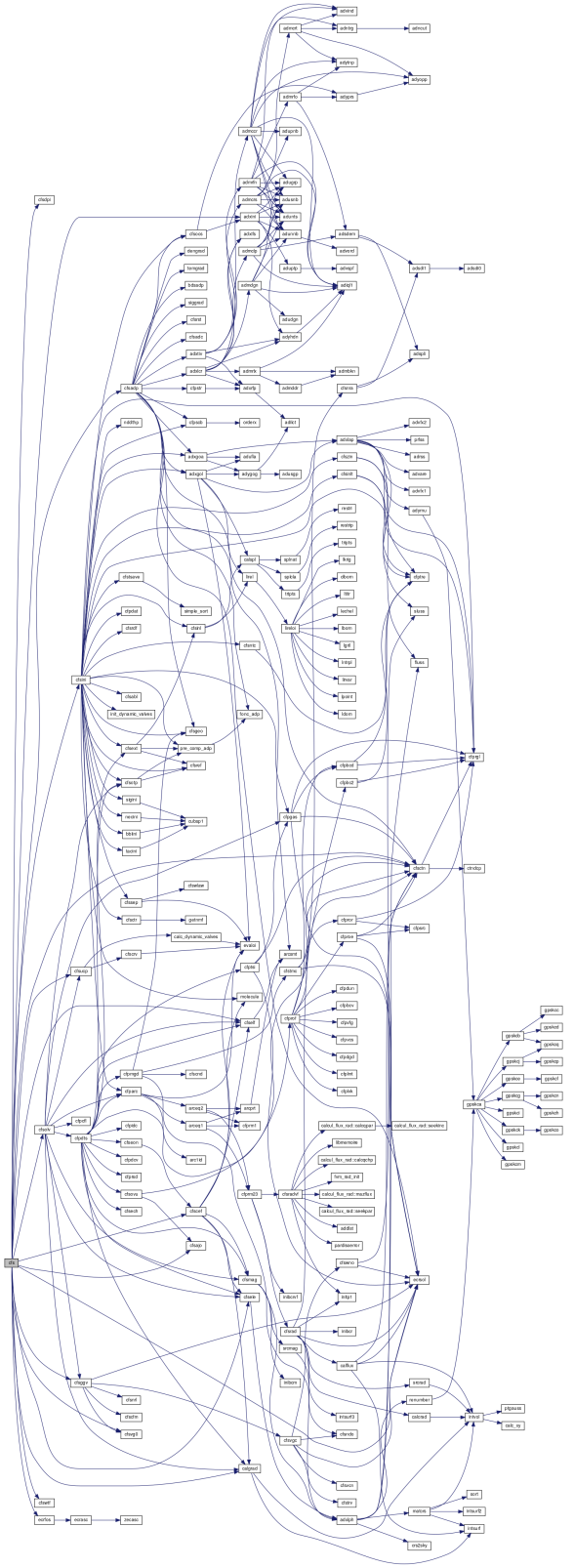


Figure 4.7 Aspect structurel de MC^3

4.4 Propositions d'améliorations et analyses des algorithmes

En corrélation avec les facteurs qui limitent les performances, on propose d'apporter de profondes modifications à MC^3 ; les propositions d'améliorations sont de natures fonctionnelles, structurelles, algorithmiques et parallèles, et mènent à des changements importants.

Pour accélérer l'exécution du programme deux points stratégiques généraux sont à considérer : réduire la quantité de travail totale et optimiser la vitesse de réalisation de ce travail. Les temps de calcul approchent un mois et on souhaite les réduire à moins d'une semaine.

Méthodologie générale de modernisation de MC^3 :

I. Créer la suite de tests de non-régression et de validation qui sera utilisée pour toutes les étapes qui suivent.

II. Nettoyer le code désuet.

III. Isoler les physiques.

IV. Identifier les points critiques à accélérer.

V. Accélérer les physiques et les points critiques un à un.

VI. Intégrer les points accélérés et réviser l'architecture (traité au Chapitre 5).

VII. Améliorer la prédiction en ajoutant un nouveau modèle (traité au Chapitre 6).

Les étapes I à III sont des phases préliminaires. Afin d'obtenir de plus forts gains potentiels d'accélération, il faut en priorité accélérer les points chauds qui représentent la plus grande part de temps CPU. Les principaux points chauds seront nommés à l'étape IV.

Le point V sera plus particulièrement développé dans la suite de ce chapitre selon les **six phases consécutives d'optimisation** :

1. Révision algorithmique et de modélisation.
2. Adaptation au problème résolu : spécialisation pour les disjoncteurs haute-tension.
3. Optimisation arithmétique, du CPI et du parallélisme d'instruction (ILP).
4. Optimisation du parallélisme de vecteur (VLP).
5. Optimisation du parallélisme de tâches (TLP).
6. Optimisation de l'efficacité mémoire.

Les 4 sections suivantes constituent la présentation des phases préliminaires de modernisation qui ont été mises en place.

4.4.1 Tests de non-régression

Les tests de non-régression sont utiles après chaque modification pour apprécier l'impact de ses changements sur les résultats. Les cas tests de validation en font partie dans notre cas. Les tests sont de différentes natures : globaux, pour vérifier le fonctionnement d'un module physique complet ou d'un calcul ; locaux, afin de tester une fonction précise ou un bloc d'instructions particulier. Parallèlement aux vérifications, des choix pertinents de tests de non-régression permettent de mesurer l'accélération de l'exécution. Ces tests sont susceptibles d'évoluer au cours des phases successives d'optimisation car ils suivent l'évolution du code source au fur et à mesure de ses modifications. Des détails supplémentaires sont fournis à la section 4.4.5 en support de la démarche d'accélération.

4.4.2 Nettoyage du code désuet

Il s'agit de la première étape à réaliser en amont de l'accélération. Du point de vue fonctionnel, on propose d'éliminer les fonctionnalités qui ne sont plus utilisées afin de réduire l'empreinte mémoire, de simplifier le code exécutable, et de **faciliter les tâches d'optimisation suivantes**.

A titre d'exemple, concernant la mécanique des fluides, les fonctionnalités apparaissant en rouge sur la figure 4.8 ont été retirées.

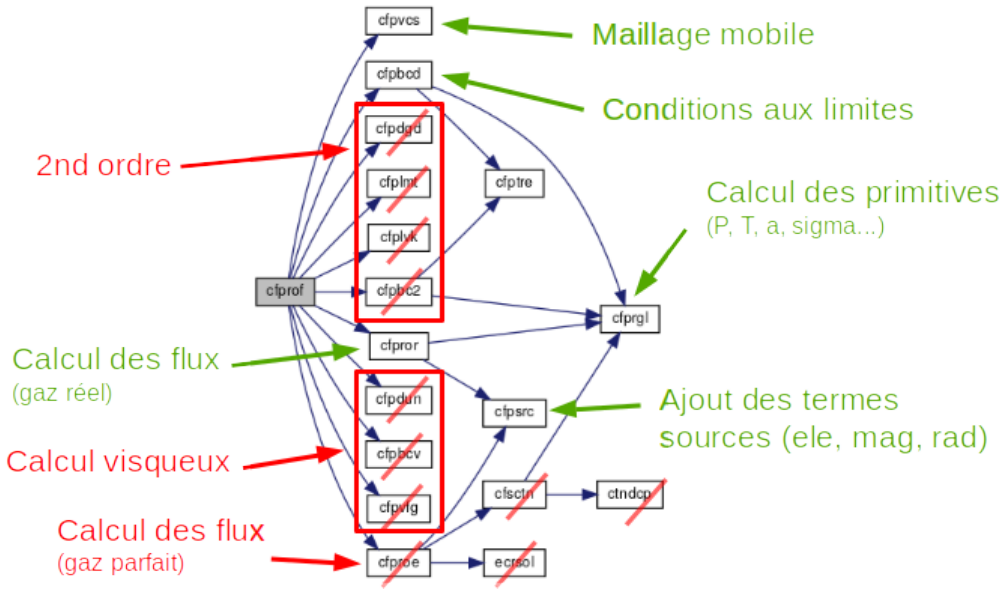


Figure 4.8 Nettoyage de code désuet - exemple mécanique des fluides

— **Solveur au 2ème ordre en espace :**
Les possibilités de calcul à l'ordre 2 en espace ne sont pas exploitées par l'industriel

et n'ont pas été maintenues à jour au fil des évolutions du programme. A court terme, on peut continuer d'utiliser le schéma à l'ordre 1.

— **Gestion des pas de temps locaux :**
Beaucoup de tableaux en mémoire et des algorithmes très demandeurs en débit d'instructions ont été éliminés. Cette région du code était responsable de la plus grande partie des limitations de performances dues au débit d'instructions, malgré une majorité d'instructions logiques très peu coûteuses en cycle CPU, donc en apparence peu onéreuses. Les pipelines étaient saturés, privant d'autres régions du code de s'exécuter dans des délais optimaux pour cause de mise en attente.

— **Calculs avec fluide visqueux, calculs avec gaz parfaits, etc :**
Les calculs en régime visqueux, compatibles uniquement avec le mode gaz parfait n'étaient plus fonctionnels mais encore partiellement présents dans des parties du code source exécutées. Les calculs en mode gaz réel sont indispensables à cause des propriétés thermodynamiques et des conditions d'utilisation des gaz de remplissage employés dans les chambres de coupure. Les portions de code concernant les propriétés de gaz parfait ont été éliminées.

Les mêmes types de nettoyage ont été réalisés sur le reste du programme et les autres physiques.

Conséquences :
Globalement, de l'espace mémoire a été libéré par le retrait des tableaux inutilisés et l'effet sur les caches de données s'est avéré bénéfique (réduction de la quantité de données à traiter et de la latence). De plus, de nombreuses branches dans le code ont été éliminées et le débit d'instructions était fortement limité dans certaines régions du code, limitant de fait d'autres régions de code pourtant indépendantes. Le tableau 4.5 résume l'impact du nettoyage sur le programme.

Tableau 4.5 Conséquences du nettoyage

thèmes	Avant	Après	Influences
Mémoire RAM	$\approx 900Mo$	$\approx 200Mo$	$\searrow$ taille du stack
Mémoire Caches	saturées en latence	latence réduite	$\searrow$ pollution, $\nearrow$ efficacité
Débit d'instructions	limitant	non limitant	$\nearrow$ vitesse, $\searrow$ taille en cache
Adressage mémoire	model = large	model = small	$\nearrow$ vitesse
Linkage	-dynamic	-static	$\nearrow$ vitesse

Le débit d'instructions n'est plus le principal facteur limitant les performances du solveur fluide de l'application MC^3. La réduction de l'empreinte mémoire et de la

quantité d'instructions lors de la suppression des fonctionnalités ont permis de réduire la taille totale maximale du code et des données en stack en dessous de 2GB. Cela permet d'utiliser des options de compilations *"static"* pour l'édition des liens et sans modèle mémoire *"large"* pour la gestion des adresses des blocs d'instructions et des données (les calculs réalisés par les **U**nités de **G**énération d'**A**dresses (**AGU**) du CPU sont simplifiés et plus rapides). Le seul nettoyage produit un facteur d'accélération de 1.11× à cette étape (des bénéfices s'appliqueront sur les tâches suivantes d'optimisation également, telle que la parallélisation). Combinées au nettoyage, les nouvelles options de compilation produisent une **accélération totale** de **1.63×** par rapport à la version originale de MC^3. A titre de comparaison, on remarque que le gain en vitesse est déjà supérieur au coefficient de 1.23× obtenu par la version parallèle fournie par le prestataire. Ceci démontre l'importance d'éliminer les sections de code qui ne sont plus pertinentes pour les utilisateurs.

4.4.3 Isolation des physiques

Telle qu'illustré sur la figure 4.9, qui représente un extrait du programme, les physiques sont fortement entremêlées. Ceci a pour conséquences : une maintenance et une évolutivité extrêmement difficile ; une optimisation des performances très ardue ; des performances en vitesse d'exécution médiocres. De plus, à moins d'une connaissance exacte des rôles de chaque fonction, le profilage est très difficile à interpréter, l'analyse d'une physique à la fois n'est pas aisée. La tâche d'isolation des différentes physiques dans des modules indépendants est aussi une tâche préparatoire à l'accélération de ces physiques. Cela permettra de se concentrer sur l'optimisation du programme dans son ensemble en intervenant physique par physique.

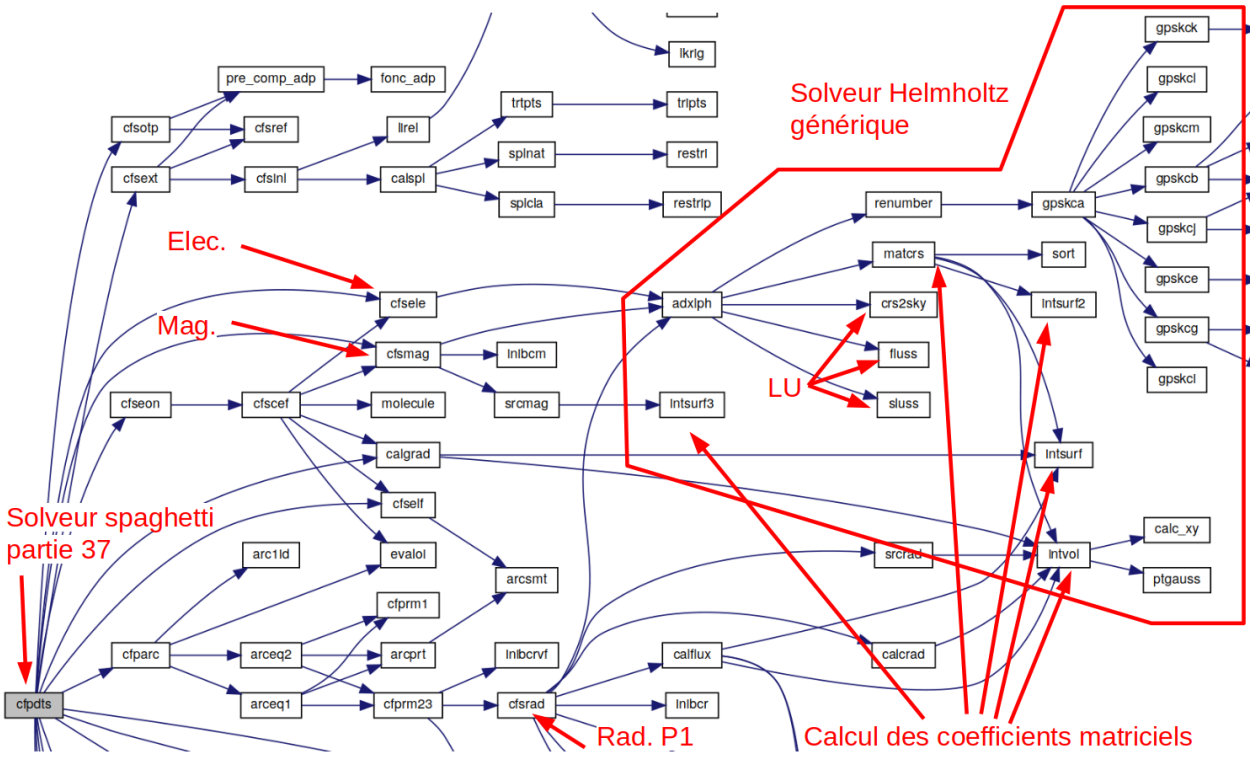


Figure 4.9 Physiques entremêlées

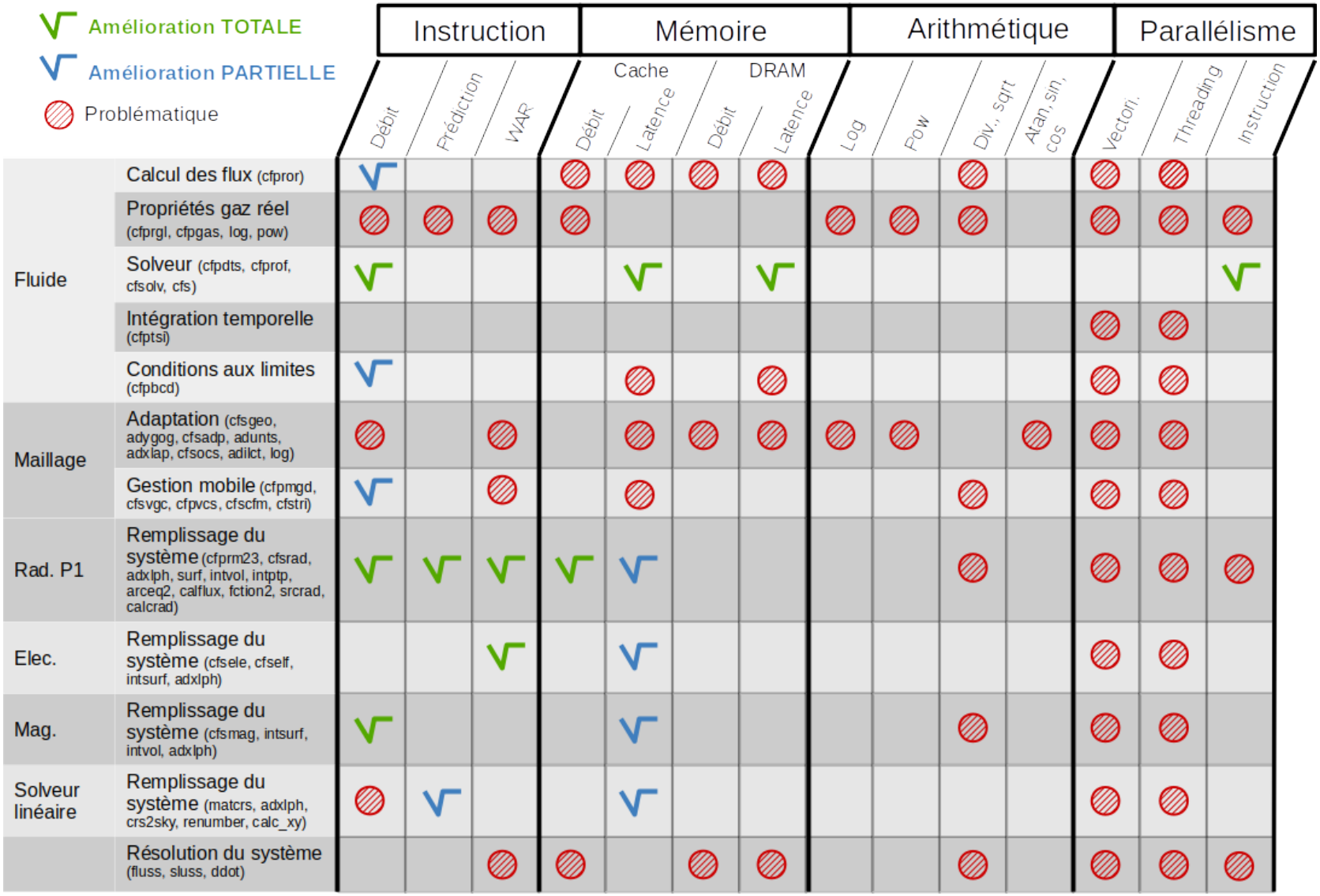
A titre d'exemple, la conception originale du solveur de Helmholtz (section 3.2.2) a été pensée pour être commune aux physiques comprenant des équations elliptiques. C'est à dire que les fonctions nécessaires à la résolution de chaque physique, notamment pour les calculs des intégrales de surfaces et de volumes sont partagées et elles contiennent des tests logiques pour orienter les opérations en fonction de chaque physique. De plus, les tableaux de données sont également communs (FORTRAN *"common"*). Finalement, les différentes physiques ne peuvent donc pas être résolues en parallèle indépendamment les unes des autres à cause de ces dépendances. Cela produit d'autres conséquences néfastes sur l'efficacité du processeur : les instructions à traiter et les branchements dans le programme sont beaucoup trop nombreux et de l'espace en mémoires caches est gaspillé par des données et des instructions qui ne seront au final pas exécutées. On a isolé par modules indépendants les fonctions pour les calculs du potentiel électrique, du champ magnétique et du rayonnement par la méthode P1. Le déroulement des instructions s'en trouve simplifié et la plupart des branches ont été supprimées rendant la suite d'instructions plus facilement prédictibles *a priori* par le processeur. Cela permet d'une part de meilleures performances en temps de calcul, et d'autre part cela facilite grandement les tâches de maintenance du code qui devient beaucoup plus explicite et lisible. Des redondances de données sont également éliminées, libérant de fait de l'espace

dans tous les niveaux de mémoire et évitant les opérations de copies inutiles. De plus, le profilage de MC^3 a démontré que ces parties du programme sont sujettes à des limitations de performances dues à la mémoire et aux débits d'instructions. Ceci provient de l'entremêlement des physiques et de leurs exécutions consécutives partielles, les mêmes tableaux étant régulièrement déchargés/chargés du/dans les caches au cours de la même itération principale ce qui cause beaucoup de trafic inutile (données et instructions) ralentissant grandement les performances.

Un solveur de Helmholtz générique était donc une erreur de conception d'un point de vue des performances. Le solveur a été découpé, certaines portions du code source sont alors dupliquées, telles que les calculs des intégrales de surface. Après la réorganisation du programme et la segmentation des physiques, **les limitations de performances en débit d'instructions ont totalement été éliminées des calculs des physiques elliptiques** et les métriques CPU révèlent une **diminution de la latence moyenne** du programme de $\approx$ **23**%. Le facteur d'**accélération** de la résolution des physiques elliptiques est très positif, par exemple pour le rayonnement P1, il est de **1.72×**.

A ce stade des opérations préliminaires, le **facteur d'accélération globale** de MC^3 approche **1.82×** et le résumé des facteurs limitant les performances de MC^3 est ainsi transformé selon le tableau 4.6.

Tableau 4.6 Améliorations des facteurs limitant les performances de MC^3



4.4.4 Identification des points critiques

Les principaux points critiques identifiés lors du profilage restant à améliorer après les étapes préliminaires présentées dans les sections précédentes sont :

- mise à jour des propriétés primitives gaz réel (P, T, a...)
- calcul des flux - mécanique des fluides
- gestion des mouvements géométriques et du maillage
- adaptation du maillage
- ordonnancement et coloriage du maillage
- calcul des coefficients matriciels et résolutions - solveur de Helmholtz

La prochaine étape consiste à accélérer les physiques et plus précisément ces points critiques car les gains de vitesse associés devraient être les plus impactant globalement.

4.4.5 Accélération des physiques et des points critiques

Les modifications apportées en phase préliminaire ont été réalisées incrémentalement sur le code source original de MC^3 et en FORTRAN 77. Dans les prochaines étapes, le FORTRAN 77 est abandonné au profit du C++ afin de bénéficier de la flexibilité et des hautes performances de ce langage et d'un environnement de développement moderne et efficace. Les six phases d'accélération, introduites à la section 4.4, induisent des changements structurels et algorithmiques successifs. Les liens entre ces six phases doivent être cohérents et une vision d'ensemble approfondie du programme est nécessaire pour minimiser le nombre de modifications inutilement répétitives et localisées dans les mêmes portions du code source. D'une manière générale, on cherche à augmenter l'intensité arithmétique, c'est à dire le ratio FLOPS/byte. Les propositions citées lors des phases qui suivent ont toutes été appliquées. Les résultats intermédiaires de performance obtenus sont présentés à la fin de chacune des six phases d'optimisation sous la forme d'un graphique, rempli incrémentalement lors de leur succession. Cela permettra de suivre, au cours des étapes successives d'optimisation, l'augmentation du facteur d'accélération par rapport à la version de référence de MC^3. Les modifications se sont appuyées sur l'expérience propre de l'auteur et s'inspirent notamment de la littérature suivante :

- Fowler et Beck [26], Fowler [27] concernant le *refactoring* de code,
- Braude [15] qui traite du design des logiciels de la programmation à l'architecture,
- l'ouvrage de McCool et al. *Structured Parallel Programming* [72] et plus particulière-

ment les chapitres concernant les modèles de design [73], la réorganisation des données [74] et le modèle *fork-join* [75],

— les chapitres de l'ouvrage sur l'architecture Intel Knights landing de Reinders [41, 99, 100, 103] décrivant respectivement des optimisations micro-architecturales, du parallélisme de tâches, de la vectorisation et des fonctions intrinsèques de vectorisation appliquées aux accélérateurs Intel Xeon Phi mais qui sont également transposables aux architectures classiques de CPU,

— les ouvrages du même auteur *High Performance Parallelism Pearls : Multicore and Many-core Programming Approaches*, regorgent d'exemples détaillés de modernisation et d'accélération de code dans différents domaine d'application [102], les chapitres 17, 18, 21 et 22 renferment de nombreuses clefs du succès des optimisations [83, 57, 40, 98, 130],

— l'ouvrage sur la librairie TBB, de Reinders [101] est également une bonne source de méthodes efficaces à s'inspirer pour l'utilisation de la librairie,

— Gross [38] traite l'anatomie d'un système de calcul parallèle, ce qui permet d'en comprendre les enjeux,

— Akhter [3] décrit dans son ouvrage le multi-threading dans sa généralité, dans un contexte où la fréquence des processeurs a cessée d'augmenter et que seule l'augmentation du nombre de cœurs permettra la poursuite de l'augmentation des performances,

— enfin, la description architecturale CPU tiré des manuels Intel [44, 45, 46, 47] et les instructions assembleurs disponibles associées apportent des informations clefs pour l'optimisation (tailles des registres, fonctionnement des caches, coût des instructions, latence...).

.

Phase 1. Révision algorithmique et de modélisation

Dans cette section, on s'appuie sur des détails de la mécanique des fluides, de la résolution du potentiel électrique et de l'adaptation géométrique de maillage de MC^3 pour illustrer les types de modifications à réaliser lors de la modernisation.

Modification 1.a : Traitement des propriétés de gaz réel et formats de stockage, partie 1

Le schéma de Roe requiert le calcul de la vitesse du son à l'interface entre les deux états voisins. Dans l'implémentation actuelle du schéma, plusieurs appels à la fonction de mise à jour des propriétés primitives de gaz réel (la sous-routine $cfprgl$) sont réalisés pour chaque côté de cellules et à chaque itération. Cette sous-routine est la plus coûteuse du programme

complet et aussi celle qui est appelée le plus grand nombre de fois. De plus, elle comporte de nombreux calculs avec des logarithmes et des exposants. Algorithmiquement, des simplifications à fonctionnalités équivalentes sont requises, l'usage de logarithmes et d'exposants est à proscrire. La sous-routine $cfprgl$ est composée à la fois d'une partie liée à l'interpolation dans les tables de données thermodynamiques et d'une partie liée aux calculs de propriétés supplémentaires comme la vitesse du son.

Afin d'optimiser la vitesse d'exécution, on propose des stratégies basées sur les thématiques suivantes :

- **Minimisation des appels :**
  Les appels de la sous-routine $cfprgl$ sont réalisés dans la fonction de calcul des flux du schéma de Roe ($cfpror$), lors du calcul du pas de temps ($cfpcfl$), pour le calcul des conditions aux limites ($cfpbcd$) et lors de la mise à jour post-intégration temporelle ($cfpgas$). Dans le cas des 3 premières sous-routines un appel est opéré pour chaque côté des éléments du maillage, pour la dernière sous-routine un appel est réalisé pour chaque triangle. On propose de stocker la vitesse du son aux cellules. Les appels dans $cfpror$ et $cfpcfl$ sont alors éliminés, un tableau est lu en lieu et place. Pour le calcul des conditions aux limites, un algorithme mieux conçu permet d'éliminer des appels à $cfprgl$ puisqu'on utilise la technique des cellules fantômes ; les propriétés sont simplement copiées et la vitesse normale est réfléchie. Il est alors fait usage de la sous-routine $cfprgl$ uniquement dans la fonction $cfpgas$ pour chaque cellule hors cellules fantômes. On a ainsi éliminé 89% du total des appels à la fonction gaz réel.
- **Optimisation des fonctions d'interpolation :**
  Malgré la minimisation de ses appels, la sous-routine $cfprgl$ demeure la fonction qui consomme le plus de cycle CPU dans le programme complet. Elle comporte de nombreux logarithmes et exposants qui sont la partie dominante. Les algorithmes d'interpolation sont intimement liés au format de stockage des données thermodynamiques du gaz et doivent donc être étudiés en corrélation.
- **Optimisation du format de stockage des données thermodynamiques :**
  Du format de stockage dépendent les algorithmes d'interpolations. On propose d'étudier différents formats de stockage et formes d'interpolation associées dans le prochain paragraphe.

Format à base de logarithmes et masses volumiques constantes

Ce format correspond à celui actuellement employé dans MC^3, on pense pouvoir l'optimiser en réduisant les calculs de logarithmes et d'exposants. Étant donné que le solveur est basé sur la masse volumique, il était naturel d'avoir un format se basant sur des propriétés stockées

à masses volumiques constantes pour faciliter les interpolations. Cependant, les propriétés (pression, température, constante du gaz...) peuvent varier de plusieurs ordres de grandeur selon la masse volumique, surtout dans la plage des plus petites valeurs $[10^{-3} - 10.0\ kg.m^{-3}]$. Ainsi, opter pour une répartition à pas logarithmiques constants permet de répartir judicieusement les points d'information et d'en utiliser un plus petit nombre pour représenter les données avec la même précision. Le but est de réduire la taille en mémoire des données thermodynamiques. Néanmoins, la recherche des intervalles dans la table et les interpolations nécessitent le calcul de logarithmes et d'exposants qui sont très coûteux en cycles CPU et en débit d'instructions. Contrairement aux unités arithmétiques plus classiques telles que les unités de multiplication ou d'addition qui sont sur chaque cœur, les unités dédiées aux logarithmes et aux exposants sont des ressources partagées. On en retrouve généralement une seule par CPU. La parallélisation ne permet donc pas d'en accélérer l'exécution. L'utilisation des logarithmes s'opèrent à différents niveaux d'imbrications tels qu'illustrés aux figures 4.10 et 4.11.

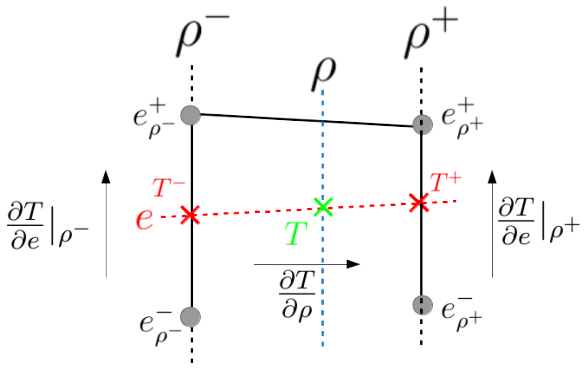


Figure 4.10 Interpolations dans les fonctions de gaz réel - étape 1

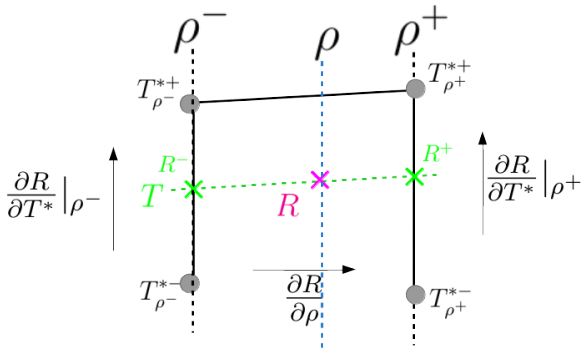


Figure 4.11 Interpolations dans les fonctions de gaz réel - étape 2

A partir de la masse volumique et de l'énergie interne, on obtient la pression et la température correspondantes à travers des interpolations dans deux sous-tables imbriquées. Les interpolations du premier niveau (Figure 4.10) permettent d'obtenir les indices pour les interpolations dans le second niveau de table (Figure 4.11). Le premier niveau est tabulé à pas logarithmiques constants en masse volumique et en énergie, tandis que le second niveau est tabulé à pas logarithmiques constants en température. Du premier niveau, on détermine la température qui permet ensuite d'interpoler la constante du gaz (R) dans la table du second niveau. La pression est alors calculée selon la loi des gaz $P = \rho\ R\ T$.

A l'initialisation de la première solution, les conditions de remplissage sont spécifiées selon

la pression et la température. Étant donné que la table citée précédemment est tabulée en fonction de la masse volumique, elle est peu adaptée aux interpolations dans le sens (P, T) -> (ρ, e).

Les tables de données thermodynamiques des gaz fournies par le laboratoire externe sont dans un format $(P-T)$. A partir de ce format, une table est générée dans le format (ρ, e) à pas logarithmiques constants et correspond à celle utilisée par le solveur. Cette étape intermédiaire nécessite beaucoup d'extrapolations en Pression et en Température pour contenir l'enveloppe de masse volumique requise, introduisant des erreurs d'approximation supplémentaires.

Pour illustrer le genre d'optimisations algorithmiques mises en place, on propose d'étudier un extrait de code source de la fonction de calcul des propriétés de gaz réel *cfprgl* présenté à la figure 4.12.

```
1
2  ! Calcul de l'indice IRHO pour la masse volumique
3  IRHO = 1 + (NBRRHO-1)*( LOG(RHO) - LOG(RHOMIN) ) / ( LOG(RHOMAX) - LOG(RHOMIN) )
4
5  ! Calcul des noeuds d'interpolation IR pour la masse volumique
6  IF ( IRHO .LE. 1 ) THEN
7    ! Extrapolation en masse volumique
8    IR(1) = 1
9    IR(2) = 2
10 ELSE IF ( IRHO .GE. NBRRHO ) THEN
11   ! Extrapolation en masse volumique
12   IR(1) = NBRRHO - 1
13   IR(2) = NBRRHO
14 ELSE
15   ! Interpolation en masse volumique
16   IR(1) = INT(IRHO)
17   IR(2) = INT(IRHO) + 1
18 ENDIF
19
20 ! Calcul des valeurs Rho pour l'interpolation en masse volumique
21 R(1) = RHOMIN * (RHOMAX/RHOMIN)**(REAL(IR(1)-1)/REAL(NBRRHO-1))
22 R(2) = RHOMIN * (RHOMAX/RHOMIN)**(REAL(IR(2)-1)/REAL(NBRRHO-1))
```

Figure 4.12 Extrait de la fonction *cfprgl* avant optimisation

Dans cet extrait de code, on prend l'exemple du calcul de l'indice en masse volumique dans la table. A partir des bornes de la table en masse volumique (RHOMIN et RHOMAX) et de la valeur d'entrée (RHO), l'indice est calculé selon la formule à la ligne *l.3* de l'extrait. Les autres indices à rechercher concernent les intervalles d'énergie interne et de température et sont déterminés à l'aide d'une formule équivalente. Cet algorithme est assez simple mais est en fait relativement mauvais en terme de performance car il nécessite les calculs de : 1 division ($\approx$ $40-100$ cycles CPU) ; 4 logarithmes népériens ($\approx$ 800 cycles CPU par logarithme) dont 3 sont pourtant constants ; 2 conversions implicites de nombre entier vers flottant ($\approx$ $4-12$ cycles CPU) correspondantes aux deux premiers nombre de l'expression ; 1 conversion implicite de nombre flottant vers entier ($\approx$ $13-17$ cycles CPU) pour le résultat

stocké dans IRHO ; 1 multiplication (= 3 cycles CPU) et 9 registres de stockage des valeurs intermédiaires. De plus, les logarithmes saturent rapidement les registres et les caches de niveau L1 (Instructions & Données) à cause des résultats intermédiaires générés. Dans le meilleur des cas et sous condition de disponibilité des ressources matérielles, l'expression coûte $\approx$ 3300 cycles CPU hors coût de stockage. Avec le support de l'analyse détaillée en annexe B et une analyse ciblée de cette partie de la fonction, on s'aperçoit qu'un seul appel de la fonction $cfprgl$ amène le processeur à saturation des ressources du point de vue du débit des instructions et du cache de données de niveau L1. L'effet est très négatif puisque les appels de $cfprgl$ sont réalisés dans des boucles composées de plusieurs dizaines voir centaines de milliers d'itérations. Les lignes *l.6* à *l.18* sont peu coûteuses mais introduisent des branches dans le code qui pourraient être évitées. Les conversions vers un nombre entier aux lignes *l.16* et *l.17* sont inutiles car déjà réalisées implicitement à la ligne *l.3*. Les lignes *l.21* et *l.22* permettent de calculer les valeurs inférieure et supérieure de masse volumique, les conversions d'entiers vers flottants, les calculs des exposants ($\approx$ $40-600$ cycles CPU) et les divisions ($\approx$ $40-100$ cycles CPU) sont très consommateurs des ressources du CPU. Dans des conditions idéales de disponibilité des ressources matérielles, l'extrait complet requiert $\approx$ 5000 cycles CPU, or, en pratique les mesures effectuées indiquent un nombre 4 à 5 fois plus élevé, indiquant une faible disponibilité des ressources CPU.

Une première possibilité d'optimisation peu efficace pourrait être d'éviter le calcul de deux des logarithmes par réduction de l'expression *l.3* ($LOG(a) - LOG(b) = LOG(a/b)$). On pourrait aussi stocker les valeurs de $LOG(RHOMIN)$ et de $LOG(RHOMAX)$ ce qui ne nécessiterait plus qu'un seul logarithme à calculer. Cependant, on sait que les données sont tabulées à masse volumique constante et que peu de valeurs sont utilisées, typiquement 7 masses volumiques différentes dans toute la table (et connues à priori). Nous pouvons en tirer partie en les stockant dans un tableau de petite taille (7) et dans lequel nous pourrons faire la recherche des valeurs supérieure et inférieure de la masse volumique donnée en point d'entrée. Ces bornes servent ensuite aux interpolations linéaires en masse volumique des autres propriétés thermodynamiques. Un extrait du code source optimisé est présenté à la figure 4.13. L'algorithme est le suivant : l'indice de la valeur inférieure à ρ est initialisé à l'avant dernière ligne du tableau, puis il est décrémenté tant que la valeur du tableau est supérieure à ρ et que le début du tableau n'est pas atteint. Les indices sont ainsi forcément bornés dans les limites du tableau (on ne fait pas d'extrapolation en masse volumique).

```
1
2   ! Calcul de l'indice IRHO pour la masse volumique
3   IR(1) = NBRRHO - 1
4   DO WHILE (RHO_LIST (IR(1)) .GT. RHO .AND. IR(1) .GT. 1)
5      IR(1) = IR(1) - 1
6   ENDDO
7   IR(2) = IR(1) + 1
8
9   ! Calcul des valeurs Rho pour l'interpolation en masse volumique
10  R(1) = RHO_LIST (IR(1))
11  R(2) = RHO_LIST (IR(2))
```

Figure 4.13 Extrait de la fonction $cfprgl$ après optimisation

Ce nouvel algorithme est particulièrement efficace pour plusieurs raisons :

— **Faibles usages des caches I & D :**
Le tableau est de petite taille et aucun résultat intermédiaire n'est stocké.

— **Réutilisation des caches :**
Au fil des appels, le tableau reste dans le cache de niveau L1, voir même mieux, dans les registres, lors de la vectorisation des boucles d'appels.

— **Coût arithmétique :**
Le coût arithmétique est dérisoire, dans le cas optimal on réalise : 2 opérations logiques (< 1 cycles CPU) + 2 addition/soustractions (< 0.3 cycles CPU) tandis que dans le pire cas 12 opérations logiques + 8 additions.

— **Prédiction de boucle et ILP parfaits :**
Depuis les processeurs Intel Pentium 4, les boucles inférieures à 16 itérations sont parfaitement prédites à l'avance par les unités de prédiction. Le coût de la recherche de l'indice en masse volumique IR(1) est quasi-nul, tout comme l'obtention des valeurs supérieure et inférieure en masse volumique.

Le nouvel algorithme consomme en pratique $\approx 6 - 12$ cycles CPU et libère également des ressources qui permettent l'accélération d'autres portions du programme.

Dans le cas des calculs d'intervalles pour l'énergie interne et la température, deux logarithmes doivent être conservés ($LOG(e)$ et $LOG(T)$) pour déterminer l'indice dans les tableaux d'énergie et de température, à moins de baser la table sur des pas linéaires. Les calculs d'exposants sont complètement éliminés grâce à la tabulation des (100) valeurs d'énergie et de température (par composition d'espèces). Finalement, 14 logarithmes sur 16 ont été éliminés et 8 exposants sur 8 ont été éliminés ce qui devrait produire la majeure partie de l'accélération de la fonctions $cfprgl$. Le coût important qui resterait à optimiser concerne le stockage des données en mémoire, trop peu efficace et qui fait actuellement usage de tableaux en 5 dimensions difficiles à optimiser. Cependant, nous préférons une approche plus globale décrite dans les prochaines lignes.

Format à base d'interpolations linéaires et pressions constantes

Les algorithmes d'interpolation précédemment décrits nécessitent la création délicate d'une table à masses volumiques constantes en échelles à pas logarithmiques. Nous proposons d'éliminer cette opération intermédiaire en utilisant les tables dans un format proche de celui originellement fourni par le laboratoire externe, c'est à dire tabulé à pressions constantes. Les intervalles de température varient à pas constant. Ce nouveau format a le double avantage de permettre une réversibilité complète des fonctions d'interpolations thermodynamiques et d'éliminer l'usage des logarithmes et des exposants et est également beaucoup plus naturel et exploitable par d'autres codes. De plus, toutes les interpolations sont linéaires, donc à faible coût. Cependant, les données sont plus volumineuses que dans le cas d'échelles logarithmiques mais elles sont stockées selon des tableaux à 1 dimension dont les accès sont beaucoup plus optimaux pour le CPU. Les données thermodynamiques sont tracées sur la figure 4.14 à pressions constantes.

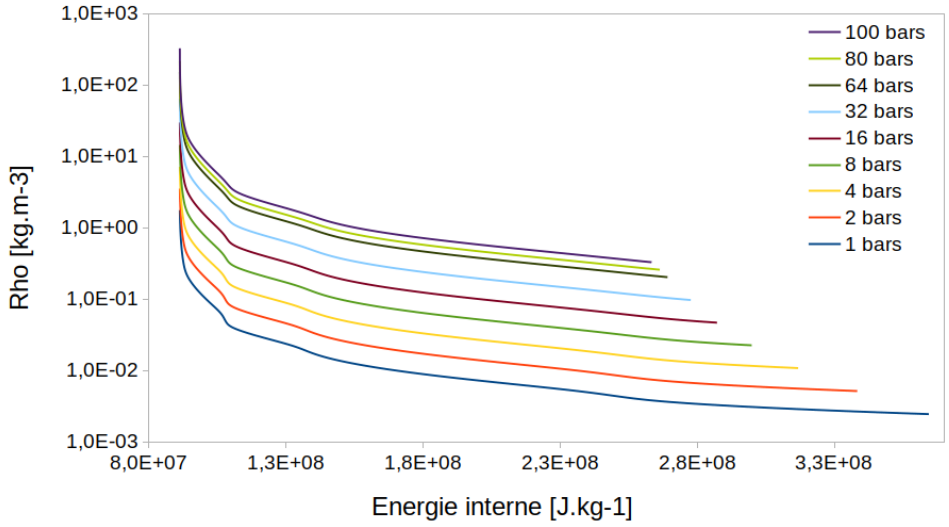


Figure 4.14 Données thermodynamiques du CO2 à Pressions constantes (représentées avec une échelle logarithmique en masse volumique)

Le schéma d'interpolation est présenté à la figure 4.15.

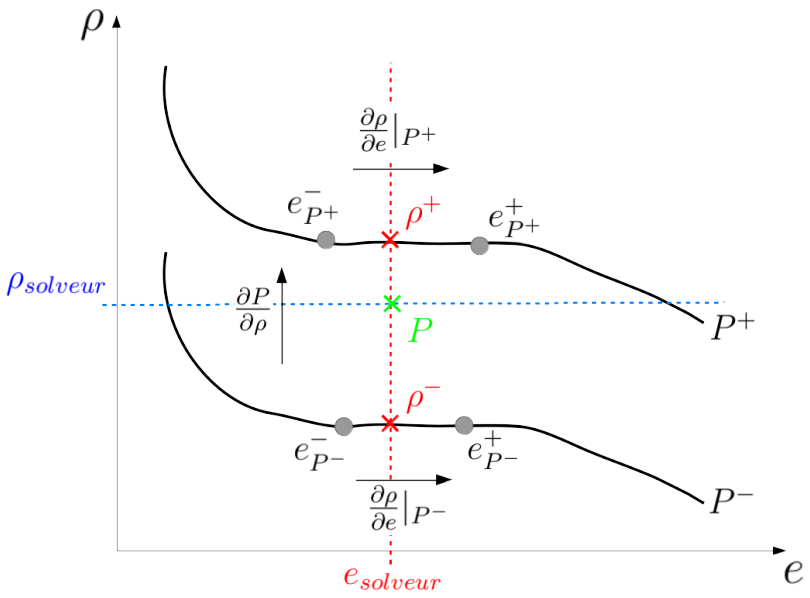


Figure 4.15 Interpolations dans le format P-T

La première étape du schéma d'interpolation consiste à déterminer les masses volumiques (ρ^- et ρ^+) à énergie interne constante aux deux bornes de pressions. On peut utiliser la pression de l'itération précédente comme candidate initiale pour accélérer la recherche de ces bornes. Si la masse volumique est supérieure à ρ^+ alors l'intervalle de pression est incrémenté, tandis que si la masse volumique est inférieure à ρ^- alors l'indice de l'intervalle de pression est décrémenté. Finalement, la pression est interpolée selon la dérivée $\frac{\partial P}{\partial \rho}$ à énergie interne constante. Les autres propriétés (température, vitesse du son et conductivité électrique) sont également interpolées en masse volumique avec les dérivées $\frac{\partial ...}{\partial \rho}$ à énergie interne constante.

Format à base d'interpolations splines et pressions constantes

Le format proposé est similaire au format précédent, cependant plutôt que de stocker les données thermodynamiques dans des tableaux à une dimension, on propose d'utiliser des splines cubiques par morceau. Les fonctions thermodynamiques sont monotones et peu oscillantes. Dans notre plage de données, il faut en moyenne 10 points de contrôle par spline. On augmente les coûts de recherche de l'intervalle dans lequel interpoler et de l'interpolation elle même mais la quantité totale de données à stocker est réduite d'un facteur très important en passant de 53 Mo pour la base logarithme à 124 Ko pour la base spline. La base de données complète peut alors être contenue dans les caches (locaux) de niveau L2 des cœurs du CPU, ce qui est un avantage indéniable pour les performances du programme. La philosophie de l'algorithme d'interpolation précédent est conservée et adaptée au format spline.

Modification 1.b : Choix des volumes de contrôle et optimisation des structures de données

Le choix du type de volumes de contrôle a un impact sur les performances mais aussi sur la solution. On peut consulter par exemple les travaux de Abanto [1] sur le sujet (appliqué à la mécanique des fluides). On bascule d'un maillage non-structuré de triangles vers un maillage cartésien pour toutes les physiques, exceptée la mécanique des fluides. Cela permet de réduire le coût de construction et de résolution tout en simplifiant le code source. Des interpolations au premier ordre entre les deux types de maillages sont réalisés pour transférer les propriétés nécessaires aux différentes physiques. Le modèle de rayonnement FVM est très coûteux, de la construction du système matriciel à la résolution. Comme il s'apparente à un lancé de rayons, le nombre d'inconnues croît rapidement avec le nombre de directions (rayons). Des maillages triangulaires d'environ 50 000 éléments génèrent alors des systèmes linéaires de plusieurs millions de valeurs non-nulles. L'optimisation des accès mémoire est donc cruciale pour ce modèle et le passage à un maillage cartésien permet d'alléger les coûts d'accès.

Concernant la mécaniques des fluides, on souhaite conserver des volumes de contrôle basés sur les triangles dans un environnement de maillage mobile *body-fitted*. Appliquée à la mécanique des fluides, la méthode du type frontières immergées (**IBM** : **I**mmersed **B**oundary **M**ethod) sur maillage cartésien a été explorée et s'est avérée très prometteuse en termes de vitesse d'exécution. Cependant, elle requiert une étude scientifique à part entière, ceci constituera des travaux futurs.

Le changement d'un maillage non-structuré vers un maillage cartésien avec frontières immergées (IBM) comportent de nombreux avantages : efficacité des caches excellente, structures de données inexistantes puisque implicites, efficacité en vectorisation et parallélisation excellente, économie de stockage importante, résolution matricielle très efficace avec des formats matriciels optimisés, simplification des algorithmes. Les calculs des gradients sont notamment simplifiés (voir Figure 4.16). En effet, lorsque les points L et R ne sont pas alignés sur la normale à la frontière des volumes de contrôle et que l'intersection i n'est pas située à mi-distance sur $[a, b]$, l'évaluation du gradient fait alors intervenir des termes reliés aux nœuds a et b pour lesquels les valeurs doivent y être interpolées (le lecteur peut consulter l'ouvrage de Versteeg [125], *section* 11.8, pour plus de détails sur le sujet). La structure de donnée est alors plus chargée d'informations et les calculs sont plus coûteux que dans le cas d'un maillage cartésien pour lequel les nœuds a et b n'interviennent pas. Les coordonnées des centres et des nœuds sont calculées implicitement donc non stockées, réduisant encore le coût des calculs des gradients (moins de données sont nécessaires dans les caches). Un traitement différent est appliqué pour les nœuds voisins aux frontières et qui ont des conditions aux limites Dirichlet, localement, les gradients sont calculés à partir des distances réelles à la parois ∂x et ∂y plutôt qu'avec les pas de mailles dy et dx. Les cellules extérieures au domaine

conservent les valeurs Dirichlet imposées à la parois par définition.

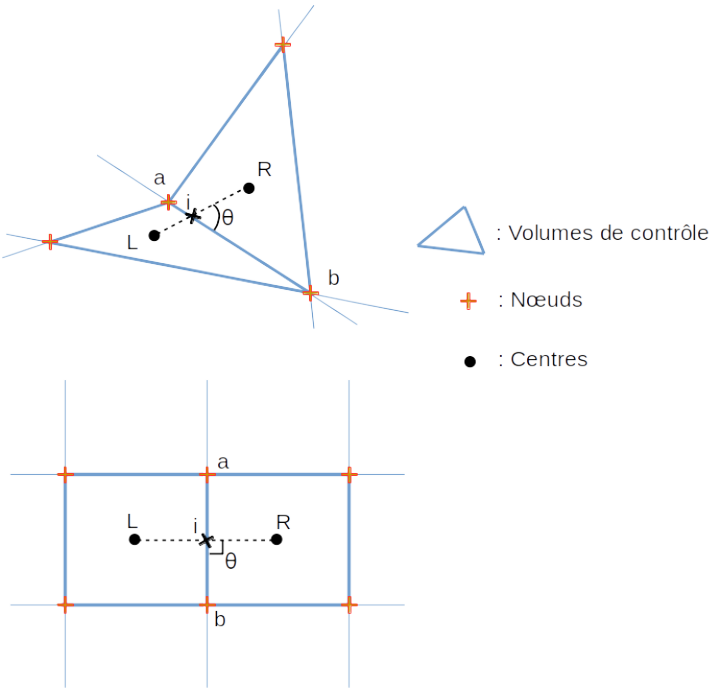


Figure 4.16 Configuration du calcul des gradients

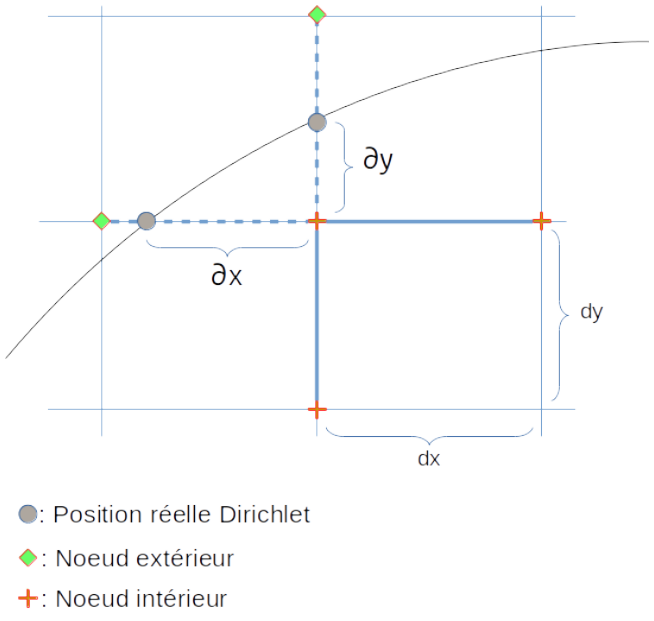


Figure 4.17 Traitement aux frontières de type Dirichlet

Le mailleur utilisé dans MC^3 génère un nombre élevé de structures de données dont une partie importante n'est pas utilisée. Mise à part l'utilisation d'une quantité de mémoire plus

importante, ces structures sont aussi maintenues à jour au prix d'algorithmes coûteux en temps d'exécution. La structure des données du maillage maintenant nécessaire uniquement aux volumes de contrôle du type triangle, appliquée à la mécanique des fluides, est grandement simplifiée et constituée de : la liste des 3 sommets, des 3 voisins et de leurs positions relatives et la liste des côtés du maillage et de leurs triangles voisins respectifs. Le coûts de maintient est beaucoup plus faible. De plus, les volumes de contrôle basés sur les nœuds sont irréguliers (voir figure 3.3), c'est à dire que le nombre de voisins est variable et leurs indices ne sont pas consécutifs. Pour conséquence, les matrices creuses ont un nombre de valeurs non nulles variable pour chaque équation, les accès aux données sont donc irréguliers également et non contigus. La renumérotation du maillage participe à l'amélioration de la proximité en mémoire, mais l'efficacité n'atteint jamais celle du maillage cartésien.

Modification 1.c : Changement de solveur linéaire et de format de stockage matriciel

Le solveur linéaire utilisé dans MC^3 est de type LU à base de matrices creuses stockées en format ligne de ciel. Ce format matriciel se prête bien à la méthode LU sans pivot telle qu'utilisée dans MC^3. L'inconvénient de ce type de solveur linéaire est son appétit très important pour la quantité de mémoire, c'est une méthode directe. De plus, les opportunités de parallélisation dans ses algorithmes sont quasi-inexistantes ce qui limite de fait les performances en vitesse de résolution. D'après les mesures effectuées (annexe B), l'implémentation réalisée dans le code source de MC^3 est peu efficace.

Dans le cas des volumes de contrôle cartésiens, qui sont maintenant utilisés pour les physiques autres que la mécanique des fluides, les indices matriciels sont implicites. On va utiliser un format matriciel adapté, inspiré du format DIA [111] pour lequel seules les valeurs et les distances relatives à la diagonale (en termes d'indices) sont stockées. Les opérations matricielles ont été optimisées pour tenir compte des particularités de ce format. Le format à base DIA possède d'excellentes propriétés d'accès aux données ce qui permet d'obtenir de très bonnes performances en vectorisation et en parallélisation. Ce nouveau format permet de se passer de la renumérotation des inconnues dans la procédure de résolution. La renumérotation permet dans le cas de MC^3 de réduire la largeur de bande des matrices qui a pour effet de « rapprocher » les données en mémoire, augmentant la probabilité de les avoir déjà chargées en cache. Les opérations matricielles sont alors accélérées par la réduction du nombre d'accès mémoire requis. La renumérotation (répétée pour chaque physique) génère beaucoup de trafic d'instructions et de données, le profilage du code a montré des pics de débits synchronisés avec les appels des fonctions de renumérotation. Au final, le gain de temps de résolution lors des opérations matricielles ne compensait pas le temps nécessaire aux opérations de renumérotation. C'était donc une étape non rentable. On renumérote plutôt le maillage, la

numérotation est alors commune pour toutes les physiques, et réutilisée de manière efficace entre les appels des fonctions d'interpolation entre maillages.

Enfin, on remplace le solveur LU par un solveur itératif de type **CG** (**C**onjugate-**G**radients) qui est compatible avec le type de matrices que nous utilisons (symétriques et définies positives). Son niveau de parallélisme est proche de 100% et sans nécessité de synchronisation inter-threads. Cependant, contrairement à LU, cette méthode requiert une solution initiale dont la qualité déterminera le nombre d'itérations à réaliser pour atteindre la convergence. Des solutions d'accélération sont proposées dans les prochaines sections. Un solveur en langage C++ a été écrit pour tenir compte de manière optimisée du format de stockage DIA, aucune implémentation commerciale ou communautaire n'est disponible.

Le profilage du solveur FVM a démontré que 99% du temps de calcul est passé dans la résolution du système linéaire (Intel PARDISO). C'est donc l'étape à optimiser en priorité pour la résolution du rayonnement avec la méthode FVM. Sachant que pour un maillage triangulaire d'environ 50 000 éléments le temps de calcul est de 20 secondes par bande et par appel (soit 100 secondes pour les 5 bandes du gaz SF6 ou 180 secondes par appel pour les 9 bandes du CO2). Les solveurs linéaires CG ou BiCGStab (**B**i-**C**onjugate **G**radients **S**tabilized) introduits pour le solveur de Helmholtz ne sont pas compatibles avec les matrices (non-symétriques et quelconques). On suggère donc de s'orienter vers un solveur linéaire du type GMRES (**G**eneral **M**inimized **RES**idual). Ce type de solveur est parallèle à presque 100% et la convergence dépend de la qualité de la solution initiale (solveur itératif) et du critère de convergence que nous avons fixé à 10^{-9} pour tous les solveurs linéaires. De plus, les formats matriciels très optimisés pour un maillage cartésien permettent d'obtenir un solveur linéaire très performant en vitesse d'exécution.

Résultats de performance de la phase 1 :

Les accélérations sont conséquentes pour toutes les physiques (Tableau 4.7), et la stratégie de choisir le bon algorithme plutôt que chercher à en accélérer un mauvais a porté ses fruits. Pour le rayonnement FVM, l'accélération est principalement due au changement de solveur linéaire.

Tableau 4.7 Facteur d'accélération à l'issue de la phase 1 par rapport à MC^3 initial

	phase 1	phase 2	phase 3	phase 4	phase 5	phase 6
Fluide	**4x**	-	-	-	-	-
Élec.	**17x**	-	-	-	-	-
Mag.	**13x**	-	-	-	-	-
Rad. P1	**14x**	-	-	-	-	-
Rad. FVM	**60x**	-	-	-	-	-
Maillage	**5x**	-	-	-	-	-

Phase 2. Adaptation au problème résolu : spécialisation pour les disjoncteurs

Dans cette section, on propose de modifier certains algorithmes afin de s'adapter au fonctionnement des disjoncteurs haute-tension. En effet, avec une connaissance approfondie du problème résolu, de nouvelles perspectives d'accélération sont possibles. Des propositions d'améliorations sont faites pour en tirer avantage. Par ces propositions, on cherche à réduire la quantité d'algorithmes dit *"brute force"* en minimisant la quantité de travail réellement nécessaire.

Modification 2.a : Détection dynamique de régions de calcul

La complexité algorithmique de la résolution matricielle ne dépend pas directement de la taille des matrices des coefficients, mais du nombre d'éléments non-nuls qu'elles contiennent.

Pour minimiser le nombre de valeurs non-nulles, on propose de réduire le nombre total d'inconnues (donc également la taille du système), en introduisant un concept de sélection de régions de calcul par physique. Par exemple, pour le calcul du rayonnement par la méthode P1, tous les nœuds du maillage compris dans la région fluide étaient pris en compte dans le système matriciel, or, les régions éloignées du cœur de l'arc ne sont pas soumises au rayonnement ou tout du moins à des niveaux négligeables. La température locale est un bon indicateur des régions actives. L'algorithme de sélection (voir Figure 4.18) est le suivant : la boîte englobante contenant les éléments à des températures supérieures à une valeur seuil est déterminée. Des marges sont alors ajoutées dans les directions azimutale et radiale pour

laisser libre la propagation de l'arc sans restriction. Les éléments concernés représentent entre <1% et 10% suivant la phase de calcul et donc l'accélération est conséquente.

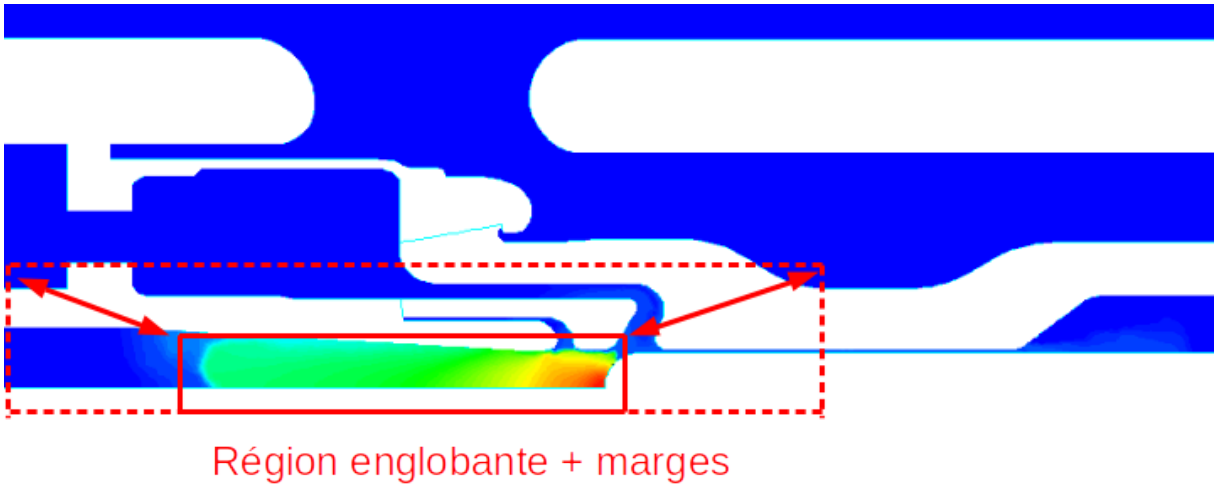


Figure 4.18 Sélection de zone active par physique - exemple pour le rayonnement P1

La détection dynamique des régions de calcul appliquée à la mécanique des fluides a aussi été implémentée : une décomposition de domaine est introduite (et réutilisée à la phase 5 d'optimisation (threading)). Le domaine est décomposé en régions temporelles dynamiquement activées en fonctions de l'écoulement, excepté pour les zones qui contiennent des parois mobiles et qui sont toujours actives. Pendant environ un million de pas de temps, c'est à dire, approximativement la moitié du nombre total d'itérations d'un calcul type, le gaz est au repos dans plus de la moitié du domaine de calcul. Dans certaines régions, le calcul de mécanique des fluides peut donc être suspendu et réactivé lorsque les flux à leurs bords dépassent une valeur seuil. D'après la direction générale des écoulements, la réactivation est définitive. En effet, les gaz ont tendance à se propager du centre de la géométrie vers les échappements et la cuve. Ainsi, le domaine de calcul fluide, s'agrandit au fur et à mesure de l'avancée d'un calcul.

Modification 2.b : Traitement des propriétés de gaz réel et formats de stockage, partie 2

Afin de favoriser les régions de données les plus utilisées et donc d'optimiser le chargement des données dans les caches, on a découpé la table globale en sous-tableaux. Cela permet ainsi de ne charger que partiellement dans les caches la table de données. Si certaines parties de l'espace de travail P-T ne sont pas utilisées alors elles ne seront pas chargées du tout. En revanche, grâce à la réduction de l'espace mémoire requis, les données encore chaudes dans les caches et qui sont fréquemment utilisées auront un plus fort potentiel de persistance. Il est donc déterminant de connaître l'usage des données dans l'espace de travail P-T. Les figures 4.19 à 4.23 présentent les résultats d'une étude statistique de l'usage des données lors d'un calcul d'arc à fort courant. Les histogrammes représentent le pourcentage de cellules en fonction des intervalles de pressions et de températures à différents instants. La région comprise dans les bornes $[5-15]$ $bars$ et $[250-1500]$ K est de loin la région la plus utilisée, l'efficacité de ces accès doit donc être maximisée. Avant la séparation des contacts électriques, la région utilisée est très restreinte (Figure 4.19), puis, lors de la séparation et de l'allumage de l'arc, cette région s'étend en température et en pression (Figures 4.20, 4.21). Lorsque le courant d'arc commence à diminuer, les températures et les pressions aux cellules baissent, la zone d'utilisation P-T se restreint de plus en plus (Figure 4.22). Finalement, à l'extinction l'intervalle de température s'est très fortement réduit. Le découpage en sous-tableaux réalisé permet de s'adapter à toutes ces évolutions dans la solution.

La taille totale de stockage des données n'a été ni réduite, ni augmentée, cependant la quantité de données simultanément stockée en caches est significativement plus petite ($\approx$ 14 Ko, suivant situation). Une accélération supplémentaire est alors produite.

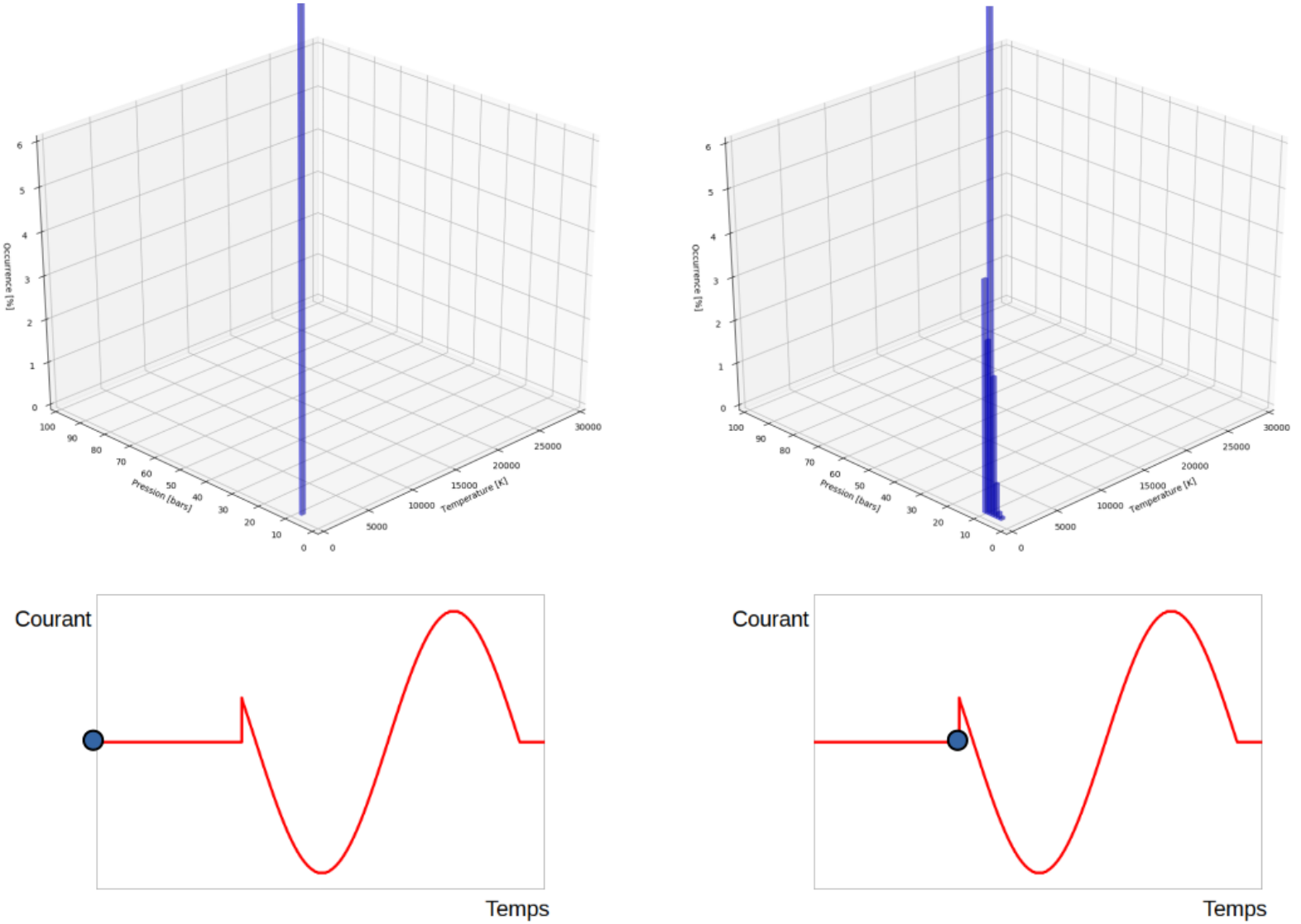


Figure 4.19 Statistiques d'utilisation P-T, a) Avant séparation

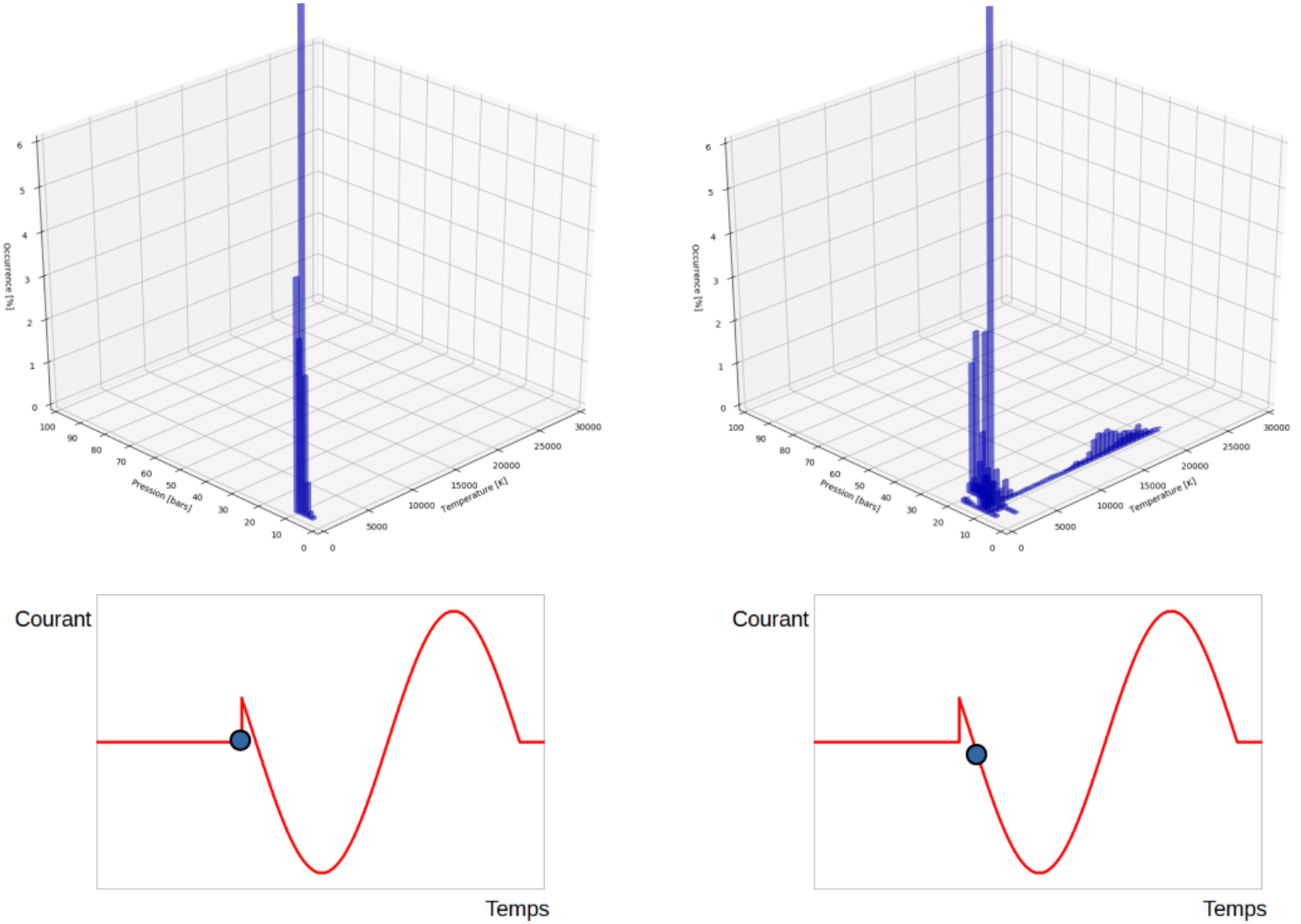


Figure 4.20 Statistiques d'utilisation P-T, b) Début d'arc

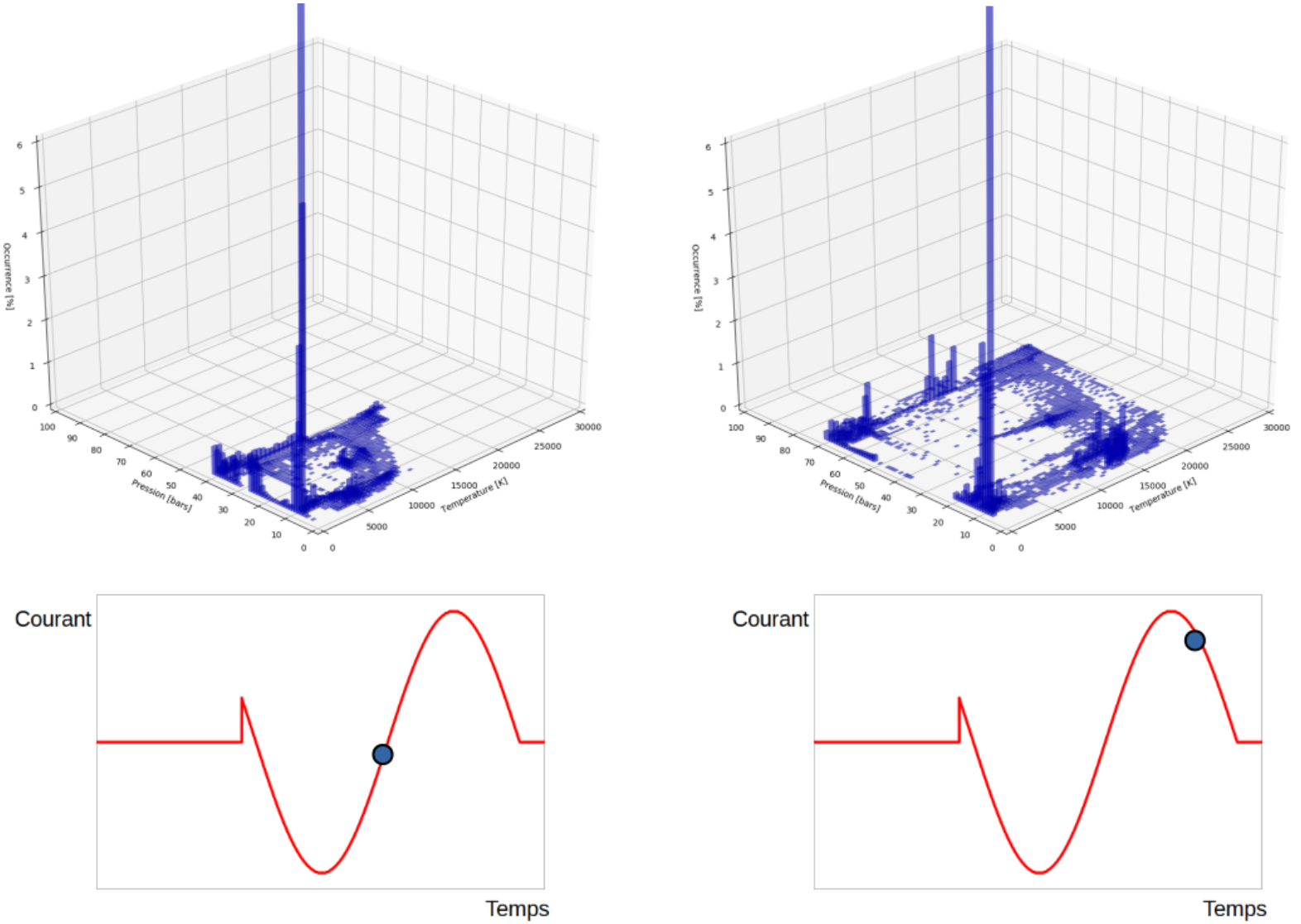


Figure 4.21 Statistiques d'utilisation P-T, c) Chauffage et montée en pression

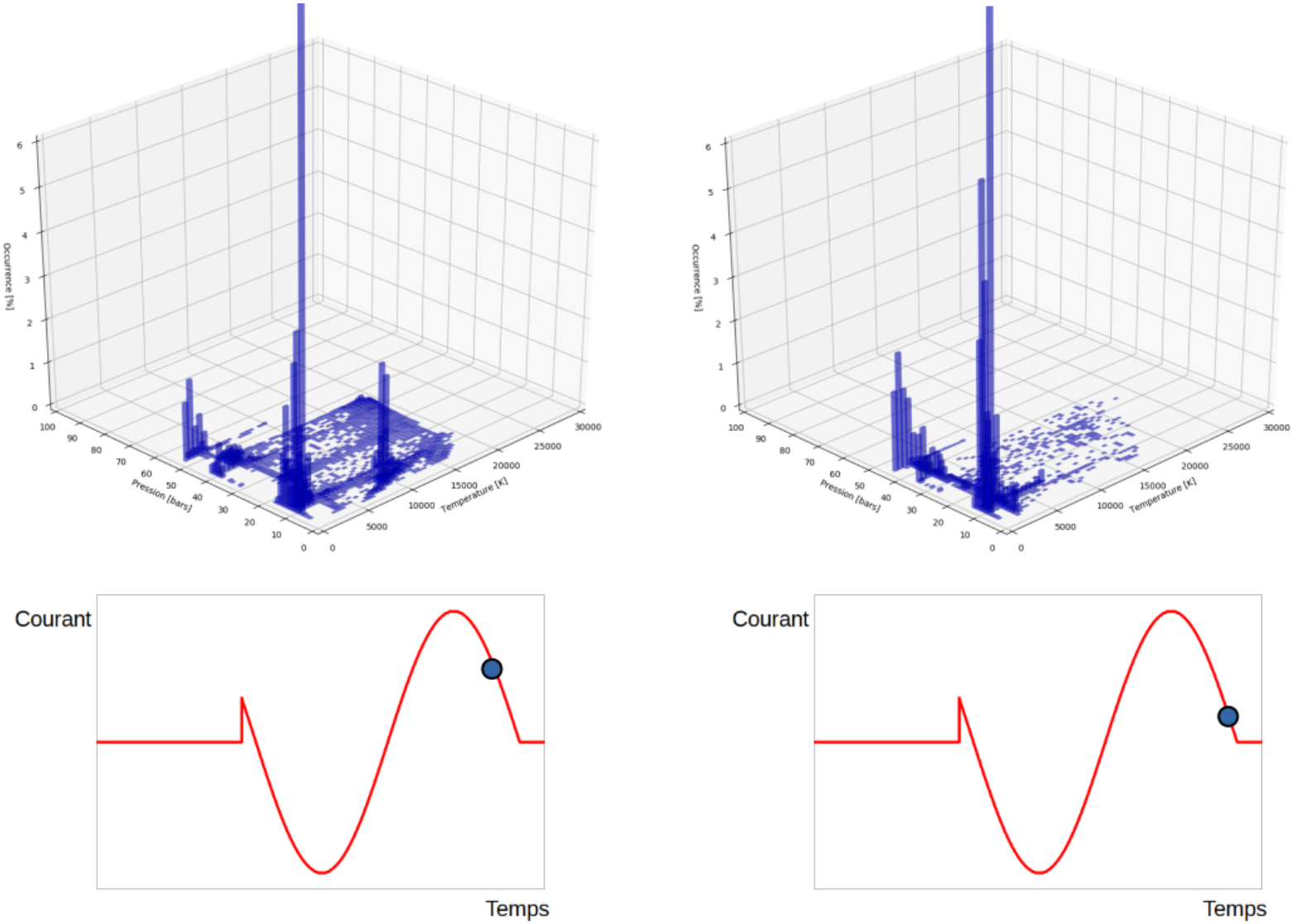


Figure 4.22 Statistiques d'utilisation P-T, d) Phase de refroidissement

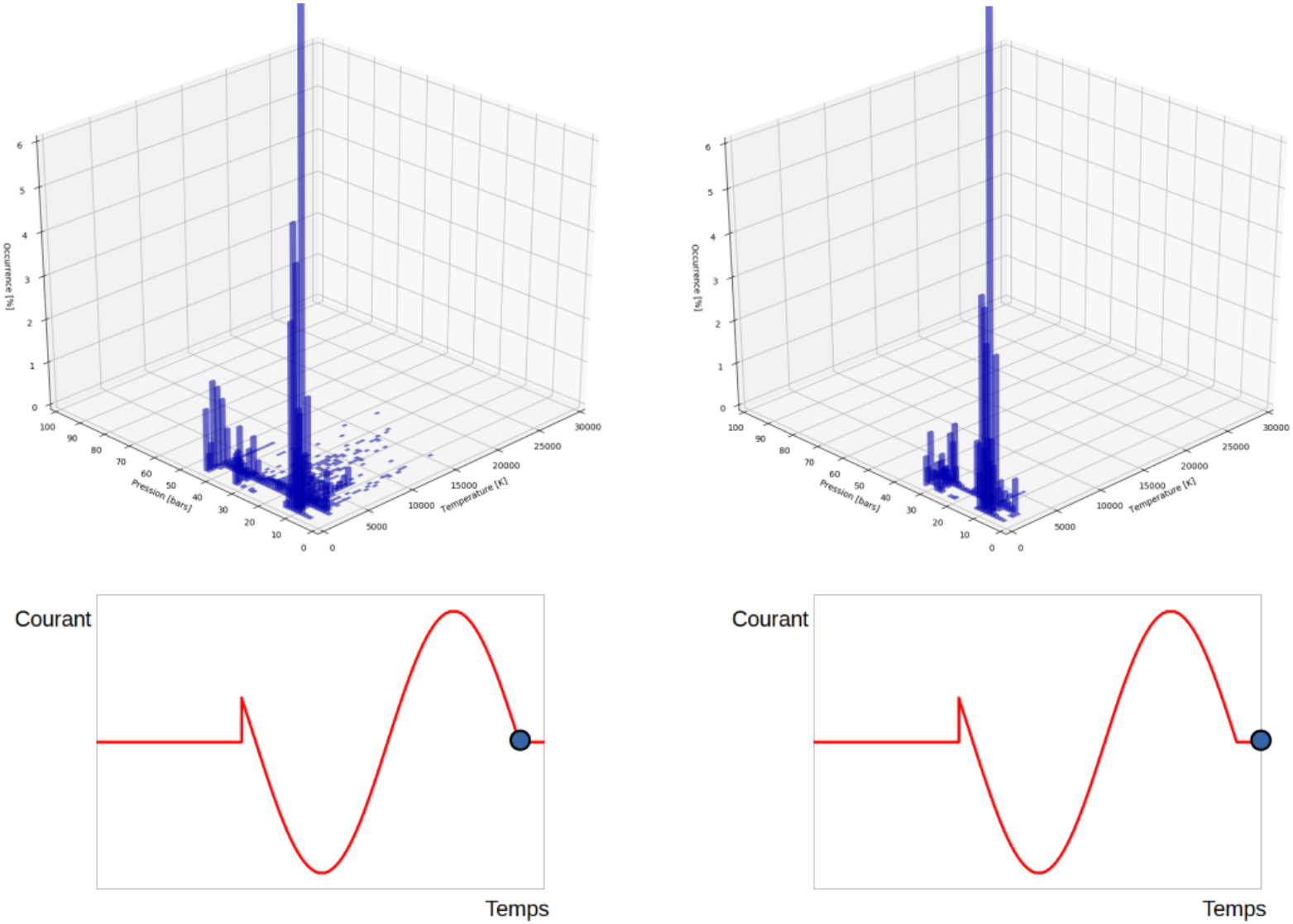


Figure 4.23 Statistiques d'utilisation P-T, e) Extinction de l'arc

Modification 2.c : Accélération de la convergence

Les performances des solveurs linéaires itératifs dépendent fortement de la solution initiale. Ainsi, de nombreuses méthodes de pré-conditionnement ont été développées pour améliorer les taux de convergence. Nous avons testé parmi les plus efficaces des méthodes développées : ILUT, ILU0, Jacobi et Row-Scaling. De par leur nature séquentielle, ces deux premiers pré-conditionneurs sont potentiellement les moins rapides. Cependant, la solution pré-conditionnée pourrait être une meilleure candidate en étant plus proche de la solution finale et ainsi réduire le nombre d'itérations requises produisant, globalement, une meilleure accélération. Les méthodes basées sur Jacobi sont des méthodes qui pré-conditionnent (divisent) la solution initiale avec les coefficients initiaux de la diagonale de la matrice. Elles sont parmi les plus efficaces, cependant elles sont moins précises à cause de l'opération de type division et elles peuvent polluer la solution (lorsque petite) en maintenant une erreur relativement haute vis-à-vis de son ordre de grandeur. La méthode de type Row-Scaling fait partie des méthodes de type Jacobi mais la solution est cette fois-ci pré-conditionnée par la norme unitaire des coefficients de chaque ligne (équation).

On a pré-conditionné la solution à la fois avec une de ces méthodes et en combinaison avec l'utilisation des solutions aux itérations précédentes. En effet, d'une résolution à l'autre (à différents instants de la simulation), la solution varie très peu, ce qui en fait une très bonne candidate pour accélérer la résolution suivante du système linéaire. Étant donné le nombre d'inconnues important et l'économie dans le nombre d'itérations pour converger, on s'attend à une accélération conséquente.

Modification 2.d : Gestion du remaillage

Le remaillage est actuellement effectué toutes les 50 itérations. Cependant, aucune étude n'a été réalisée pour optimiser l'intervalle entre deux remaillages.

On propose d'augmenter cet intervalle d'un facteur de l'ordre de 10. Les vitesses maximales des parois mobiles sont de l'ordre de 7 à 13 $mm.ms^{-1}$ et les pas de temps caractéristiques sont de l'ordre de 10^{-5} ms en phase d'arc et de 10^{-4} ms en phase froide, ce qui implique des déplacements d'environ 5×10^{-2} mm à 5×10^{-3} mm entre deux remaillages espacés de 50 itérations. On peut donc augmenter et automatiser sans grande contrainte les intervalles entre deux remaillages. L'avantage est l'économie de temps et de ressources puisque l'on évite la reconstruction des structures de données et les mises à jour qui sont fortement consommatrices de débit d'instructions et de débit mémoire, tout en souffrant de limitations liées à une latence importante (accès fortement aléatoires et beaucoup d'algorithmes (naïfs) de recherche peu efficaces). La déformation des mailles reste prise en compte à chaque pas

de temps de calcul fluide pour ne pas générer des oscillations dans la solution.

La détermination du seuil de la taille requise pour les triangles dépend des critères d'adaptation. Ces critères sont reliées aux gradients dans la solution et à la géométrie (courbures locales, proximité entre les parois...). Concernant les critères géométriques, à partir des tailles aux frontières, un système linéaire est construit pour déterminer les tailles au milieu du domaine. Le système est alors résolu par une méthode LU (sous-routine *adxlap*). Cette étape représente 24% du temps total de résolution. L'optimisation du calcul des physiques a pour conséquence d'augmenter significativement cette proportion passée dans l'adaptation du maillage qui représente alors la part majoritaire du temps de calcul.

On a remplacé la résolution exacte (LU, *adxlap*) par une méthode approchée basée sur une approximation des gradients (moyennages barycentriques itératifs). Obtenir une solution similaire à 95% avec une méthode beaucoup plus rapide en temps de calcul est une opportunité que nous saisissons. D'autant plus, qu'on ne recherche pas un degré de précision élevé. La dernière partie du programme qui souffrait de limitations de performance dues au débit d'instructions pourrait être éliminée par le remplacement de la sous-routine *adxlap*.

On propose également d'utiliser des maillages plus réguliers, plus uniformes en tailles et de tailles moyennes plus petites que l'usage actuel. En effet, les réglages actuellement utilisés par l'industriel sont plutôt grossiers avec des raffinements locaux très importants dans les régions de fort gradients (voir Figure 4.24). Cela ne confère pas de bonnes propriétés de stabilité et de robustesse au solveur. Les tailles requises seront déterminées avec des méthodes heuristiques lors de la phase d'industrialisation. De plus, on sait par expérience que les tailles requises pourraient être déterminées par l'utilisateur *a priori* par région, permettant de s'abstenir des calculs des gradients dans la solution aux fins d'adaptation. De plus, on souhaite introduire le concept de région de contrôle du maillage pour permettre la prise en compte de tailles préconisées pour chaque région et de dissocier celles qui nécessiteront des mises à jour liées aux mouvements des frontières et aux adaptations de celles qui seront statiques. Des gains de performances seront alors attendus puisque dans MC^3 toutes les propriétés géométriques des triangles étaient systématiquement recalculées de partout dans le domaine, y compris dans les zones où le maillage était resté statique.

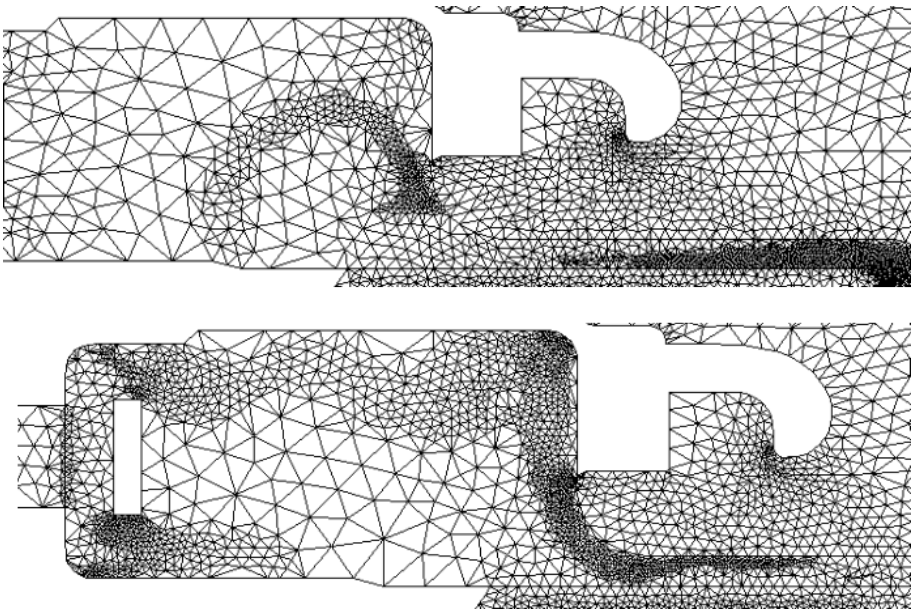


Figure 4.24 Extrait d'un maillage adapté lors d'une simulation sur géométrie réelle

Résultats de performance de la phase 2 :

La connaissance du fonctionnement des disjoncteurs a permis notamment de proposer une activation dynamique des régions de calcul pour toutes les physiques. Il s'agit d'une économie de quantité de travail réalisée à chaque itération (la taille du problème diminue) et permet d'accélérer d'un facteur additionnel très élevé (Tableau 4.8), la vitesse d'exécution du rayonnement P1 est par exemple triplée par rapport à la phase précédente d'optimisation.

Tableau 4.8 Facteur d'accélération à l'issue de la phase 2 par rapport à MC^3 initial

	phase 1	phase 2	phase 3	phase 4	phase 5	phase 6
Fluide	4x	**8x**	-	-	-	-
Élec.	17x	**17x**	-	-	-	-
Mag.	13x	**18x**	-	-	-	-
Rad. P1	14x	**47x**	-	-	-	-
Rad. FVM	60x	**237x**	-	-	-	-
Maillage	5x	**12x**	-	-	-	-

Phase 3. Optimisation arithmétique, du CPI et du parallélisme d'instruction

Dans cette section, on montre comment le CPI peut être amélioré. Il montre le nombre de cycles par instruction retirée, c'est donc un indicateur d'un manque d'optimisation potentiel, surtout si la fonction concernée est un point critique. Ainsi, plus il est faible, plus la vitesse d'exécution est élevée. Pour améliorer le CPI, un changement d'algorithme peut être nécessaire. A titre d'exemple, une modification est introduite concernant la gestion des déplacements des courbes et des mailles localisées sur les parois mobiles. En effet, le profilage a démontré qu'il s'agissait d'une fonction réalisant un grand nombre de cycles total pour un taux CPI élevé. Les opérations coûteuses, comme les divisions, peuvent conduire à des valeurs de CPI élevées lorsqu'elles sont trop nombreuses dans une certaine zone de code et que le temps CPU associé ne peut pas être masqué par d'autres instructions. Des modifications ont été implémentées pour réduire l'impact arithmétique sur la vitesse de résolution.

Modification 3.a : Nouvelle gestion des déplacements des courbes et des mailles

La gestion du domaine géométrique lors des déplacements est traitée courbe par courbe. C'est à dire que les courbes sont déplacées indépendamment les unes des autres selon les deux axes de coordonnées. Les nœuds du maillage placés sur les intersections entre courbes y sont verrouillés. Il faut donc recalculer la position des intersections après déplacement pour les y replacer. Cette tâche représente 5% dans l'implémentation actuelle de MC^3. Cependant, après accélération des autres parties du programme, cette tâche sera majoritairement limitante. L'algorithme en place est naïf, des boucles imbriquées balayent tous les segments qui composent les courbes et testent les intersections éventuelles avec tous les autres segments de toutes les autres courbes du domaine. La complexité est supérieure à $O(n^2)$.

On gère maintenant le domaine géométrique d'une autre façon. Après les étapes de prétraitement de la géométrie, les frontières du domaine de calcul sont définies comme étant des polygones composés uniquement d'une liste de sommets (voir Figure 4.25, *partie haute*). Les déplacements selon les deux axes de coordonnées sont indépendants et aucune rotation n'est réalisée lors de la simulation des chambres de coupure. Les nouvelles positions sont donc facilement mises à jour par deux additions (une pour chaque direction). La recherche des segments et les calculs d'intersections, coûteux, non robustes et peu précis à cause des erreurs d'arrondis sont éliminés (la sous-routine $cfstri$ est supprimée). Même en cas de parallélisation de la fonction, les performances sont limitées à cause des besoins de synchronisation (temps de calcul élevé, scalabilité faible et taux d'occupation CPU médiocre). Le nouvel algorithme de gestion est également 100% parallèle, chaque position de nœuds pouvant être mise à jour indépendamment et de manière contiguë par bloc de nœuds (très efficace pour les caches et la

vectorisation). De plus, il ne souffre pas de problème de précision numérique. Par expérience, il était fréquent dans MC^3 de *perdre* l'intersection entre deux courbes, le nœud du maillage n'était alors plus contraint à sa position imposée.

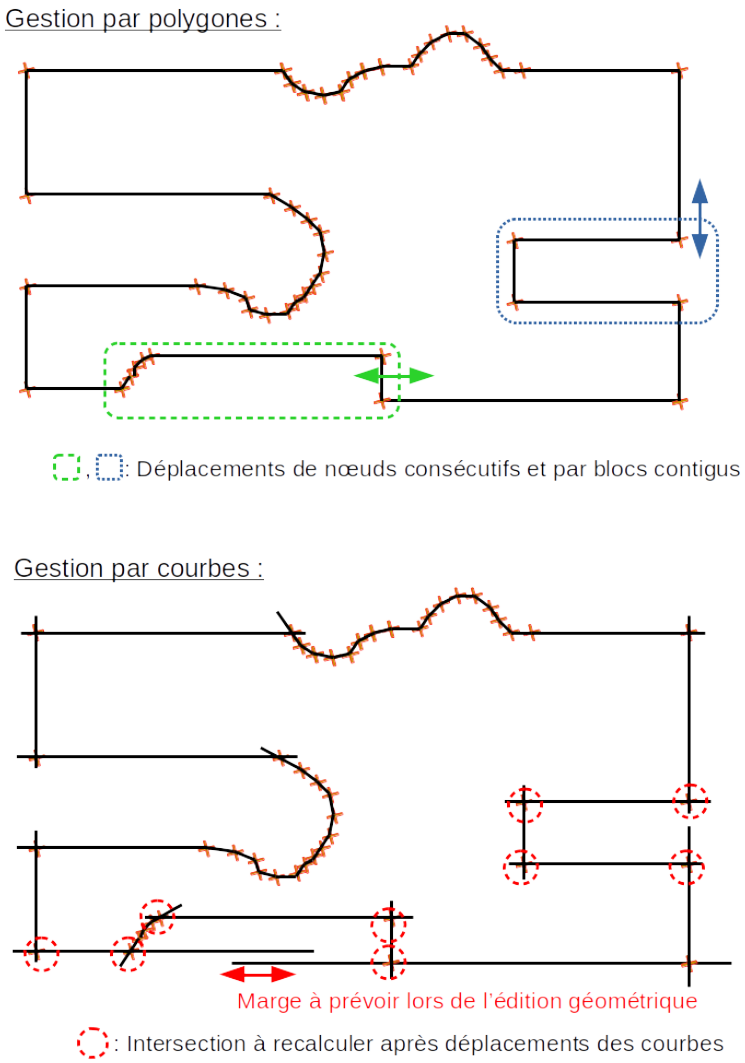


Figure 4.25 Gestion polygonale des frontières

Dans la nouvelle implémentation, le maillage peut être déplacé sur les frontières aussi bien par une méthode de type **RBF** (**R**igid **B**ody **F**unctional) où les nœuds du maillage sont solidaires du mouvement de la frontière que par une méthode de glissement (Figure 4.26). Les nœuds sont indépendants des intersections entre courbes qui ne sont plus considérées. MC^3 utilise une méthode de glissement par défaut dans laquelle des nœuds sont contraints aux intersections dont les positions sont à recalculer à chaque pas de temps.

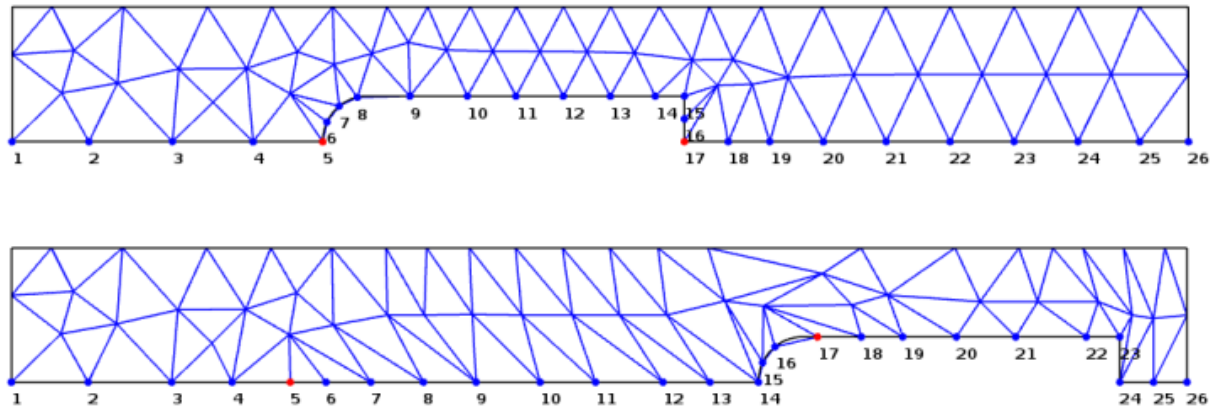


Figure 4.26 Application du glissement du maillage, sans contrainte aux intersections

Modification 3.b : Intensité arithmétique

Les instructions nécessitant un grand nombre de cycles CPU induisent des CPI élevés. C'est le cas des divisions et des calculs trigonométriques (log, sinus, cosinus...) de manière générale. Une optimisation importante visant à limiter ce type d'instructions peut apporter des gains de performance. Par exemple, le remplacement de divisions par des multiplications a été réalisé dans le code source, tel qu'illustré à la figure 4.27 dans l'exemple 1 et 2.

```
1
2       ! Exemple 1.
3       ! extrait original
4       rho = DENPLG(I)
5       u   = rhoU / rho
6       v   = rhoV / rho
7       E   = rhoE / rho
8
9       ! remplacé par:
10      inv_rho = 1.0 / DENPLG(I)
11      u   = rhoU * inv_rho
12      v   = rhoV * inv_rho
13      E   = rhoE * inv_rho
14
15      ! Exemple 2.
16      DO I=1, NPLG
17              DEDPLG(I) = DEDPLG(I) / cellVolumes(I)
18              DEUPLG(I) = DEUPLG(I) / cellVolumes(I)
19              DEVPLG(I) = DEVPLG(I) / cellVolumes(I)
20              DEEPLG(I) = DEEPLG(I) / cellVolumes(I)
21      ENDDO
22
23      ! Après remplacement
24      DO I=1, NPLG
25              DEDPLG(I) = DEDPLG(I) * inv_cellVolumes(I)
26              DEUPLG(I) = DEUPLG(I) * inv_cellVolumes(I)
27              DEVPLG(I) = DEVPLG(I) * inv_cellVolumes(I)
28              DEEPLG(I) = DEEPLG(I) * inv_cellVolumes(I)
29      ENDDO
```

Figure 4.27 Remplacement de divisions par des multiplications équivalentes

Dans l'exemple 1 (Figure 4.27), les divisons sont remplacées par des multiplications par une variable locale (*inv_rho*, *l.10*). On a ainsi économisé deux divisions et ajouté trois multi-

plications, l'économie en nombre de cycles CPU représente environ 66 %, répétée à chaque nouvelle exécution. Dans l'exemple 2, on a choisi de stocker les inverses du volume des cellules dans un tableau et de les relire. Localement, lors de l'exécution du bloc d'instruction, la quantité de données à charger depuis la mémoire reste identique, cependant, le nombre de cycles CPU nécessaires à la partie arithmétique est approximativement dix fois moins élevé. La généralisation de ce genre de modifications à tout le code source permet des gains de vitesse importants, surtout lorsque les inverses de variables sont pré-calculables et réutilisées entre plusieurs sections de code (avec des multiplications équivalentes).

Résultats de performance de la phase 3 :

La partie maillage était limitée par un CPI élevé dû aux calculs des positions des nœuds contraints aux intersections dont l'algorithme approchait une complexité de $O(n^3)$. Un algorithme moins naïf (en $O(1)$) avec une gestion polygonale des frontières règle le problème plutôt qu'une parallélisation *"brute force"* des boucles avec des conditions de réduction. Les autres physiques n'étaient pas sujettes à ce type de limitations et ne sont pas spécialement accélérées (Tableau 4.9).

Tableau 4.9 Facteur d'accélération à l'issue de la phase 3 par rapport à MC^3 initial

	phase 1	phase 2	phase 3	phase 4	phase 5	phase 6
Fluide	4x	8x	**8x**	-	-	-
Élec.	17x	17x	**17x**	-	-	-
Mag.	13x	18x	**18x**	-	-	-
Rad. P1	14x	47x	**47x**	-	-	-
Rad. FVM	60x	237x	**237x**	-	-	-
Maillage	5x	12x	**36x**	-	-	-

Phase 4. Optimisation du parallélisme de vecteur (VLP)

Dans cette section, des modifications sont apportées au code source pour augmenter le taux de vectorisation aux endroits qui importent le plus, élément essentiel pour l'obtention d'un facteur d'accélération important.

Modification 4 : Vectorisation explicite et langage assembleur

Au niveau du parallélisme, on a opté pour une vectorisation systématique de toutes les boucles, élément crucial dans l'optique d'obtenir un niveau de performance élevé avec le matériel informatique moderne. Une révision des algorithmes est nécessaire en complément pour réduire ou éliminer les goulots d'étranglements et les synchronisations entre opérations. La vectorisation permet de réduire la quantité d'instructions pour effectuer un travail identique sur de multiples jeux de données : c'est une forme de parallélisme dite SIMD (**S**ingle **I**nstruction **M**ultiple **D**ata). De l'espace est économisé dans les caches, et cela simplifie la chaîne d'instruction générée par le scheduler du CPU, libérant ainsi des ressources pour d'autres instructions (ILP). Ceci est utile pour accélérer les boucles, les mêmes instructions sont alors répétées sur un espace itératif plus court. Lorsque les données utilisées ensembles (spatialement dans la solution) sont stockées ensembles en mémoire, l'efficacité des caches et des accès mémoire pour les unités vectorielles est maximisée. Pour augmenter encore l'efficacité parallèle et vectorielle, le code doit éviter au maximum les branchements ($if - elseif - else...$) et les réductions dans les boucles ($sum = sum + a[]...$). Les interruptions système ($write$, $stop$,$send/receive...$) et les sauts ($goto$, $call...$) interdisent toutes formes de vectorisation. Le compilateur tente d'analyser les opérations réalisées pour générer les instructions les plus efficaces en temps de réalisation, dont des instructions vectorielles. Cependant, devant la complexité des algorithmes et le manque d'informations vis à vis de l'ambiguïté d'interprétation du code source, le compilateur n'est pas à même de les générer dans le cas de MC^3 sauf à quelques exceptions près (voir annexe B). Comme l'impact est critique sur les performances, il a fallu aider le compilateur à les générer. Plusieurs choix étaient envisageables : la vectorisation guidée à l'aide de directives à insérer en entête des boucles, comme par exemple $C\$OMP\ SIMD$ avec la bibliothèque de parallélisation $OpenMP$; l'insertion de conseils dans le code concernant les variables et les dépendances qui permettent la clarification du code pour le compilateur ; l'usage direct du langage assembleur pour écrire exactement les instructions à réaliser par le processeur sans interprétation du compilateur et l'emploi de fonctions (assembleur) intrinsèques de vectorisation du compilateur, à la manière de Liu *et al.* [66]. Nous avons choisi cette dernière, l'assembleur permettant d'obtenir les performances les plus importantes. La vectorisation est particulièrement intéressante pour les boucles les plus coûteuses puisqu'elle y apporte les plus grands gains de vitesse.

Résultats de performance de la phase 4 :

La vectorisation participe à réduire la saturation en débit d'instructions, puisque avec des vecteurs de longueur huit, théoriquement huit fois moins d'instructions sont délivrées aux unités d'exécution pour réaliser le même travail. Les parties du code qui sont limitées par la mémoire ou les caches ne sont pas accélérées par la vectorisation ou très peu, ce qui est le cas pour les physiques elliptiques à ce stade des modifications (Tableau 4.10). C'est la mécanique des fluides qui bénéficie le plus de la vectorisation : les boucles de mises à jour des propriétés primitives ($cfpgas$, $cfprgl$, $cfptsi$) ou du calcul des flux ($cfpror$), pour lesquelles tous les sauts, branches, réductions et interruptions système ont dû être éliminés, en ont fortement bénéficié. Sur un cœur CPU, le taux d'utilisation des unités vectorielles a progressé, il atteint environ 87% , contre seulement 2% initialement. Le langage C++ s'est révélé un élément essentiel pour la réussite de la phase de vectorisation.

Tableau 4.10 Facteur d'accélération à l'issue de la phase 4 par rapport à MC^3 initial

	phase 1	phase 2	phase 3	phase 4	phase 5	phase 6
Fluide	4x	8x	8x	**14x**	-	-
Élec.	17x	17x	17x	**19x**	-	-
Mag.	13x	18x	18x	**20x**	-	-
Rad. P1	14x	47x	47x	**49x**	-	-
Rad. FVM	60x	237x	237x	**239x**	-	-
Maillage	5x	12x	36x	**39x**	-	-

Phase 5. Optimisation du parallélisme de tâches (TLP)

L'ajout de la vectorisation seule apporte un gain significatif en vitesse. On peut obtenir un facteur multiplicatif additionnel par l'ajout de threads est crucial.

Modification 5 : Parallélisation explicite à base de tâches

En complément de la vectorisation, on propose d'opter pour l'ajout de threads systématiques de toutes les boucles. Une révision des algorithmes est nécessaire en complément pour réduire ou éliminer les goulots d'étranglements et les synchronisations entre threads. La bibliothèque TBB est utilisée pour paralléliser les tâches de travail. Cette bibliothèque est basée sur des tâches plutôt que sur des threads. C'est à dire que les threads sont initialisés au lancement du programme puis des tâches issues d'une liste principale leurs sont assignées, ainsi l'espace parallèle correspond à tout le programme. Par opposition, la librairie OpenMP crée/détruit les threads à chaque entrée/sortie dans une région parallèle ; ils se partagent le travail à l'entrée et se rejoignent à la sortie de la région parallèle qui peut être une simple boucle (modèle "*fork-join*"). Par conséquence, les coûts associés aux créations et destructions des threads doivent être amortis par une quantité de travail suffisamment importante. Dans le cas des tâches, leurs créations sont très légères et la conception du programme est plus facile à imaginer en terme de tâches indépendantes à distribuer à des travailleurs. Le scheduler de TBB se charge de la distribution des tâches et favorise les affinités vers les cœurs au fil des appels.

Dans le but d'augmenter encore l'efficacité du parallélisme de threads que nous avons ajouté à MC^3, on a réutilisé la décomposition de domaine (modification 2.a), dont chaque partie constituante est envoyée à des cœurs/CPU différents (en local sur multi-sockets de type NUMA ou distants sur nœuds de calcul inter-connectés en réseau). La décomposition de domaine permet de renforcer le lien entre proximité géométrique et proximité en mémoire. De plus, la politique d'initialisation mémoire des CPU avec des systèmes d'exploitation courants utilise la politique dite du "*first-touch*", c'est à dire que l'allocation mémoire est effectivement réalisée au premier accès à l'adresse par le processeur qui en fait la demande et non pas à la position dans le code source ni à l'initialisation du programme. Nous proposons donc d'organiser les algorithmes pour que les tout-premiers accès soient réalisés par les processeurs qui utiliseront ensuite les données. La combinaison avec des modifications précédentes telles que le format des bases de données prend tout son sens : un processeur (ou un cœur de processeur) réalisant des opérations dans la partie à haute température et haute pression utilisera des données en mémoires locales sans polluer les autres processeurs opérant dans des régions plus froides donc avec des données différentes. Cela permet de maximiser également la réutilisation des données en caches qui sont de nature plus compacte. De plus, le lancement

et la répartition des tâches dépendent du contrôle dynamique du scheduler TBB, combinés à la détection dynamique des zones temporellement actives (modification 2.a).

L'isolation du solveur de Helmholtz par physique laisse place à l'augmentation de la scalabilité. En effet, chacune des physiques est vectorisée et parallélisée selon les mêmes principes évoqués plus haut pour le module de mécanique des fluides mais elles peuvent en plus maintenant être résolues indépendamment les unes des autres sur des processeurs/cœurs/nœuds différents. Ainsi, le facteur d'accélération maximal est augmenté et se projette sur un nombre de cœurs plus élevé, la scalabilité est améliorée. De plus, les opérations et les formats matriciels ont été optimisés pour maximiser les performances de la vectorisation et de la parallélisation.

Résultats de performance de la phase 5 :

Malgré les limitations liées à la mémoire et aux caches, la parallélisation accélère l'exécution (Tableau 4.11). Utiliser plusieurs cœurs du CPU permet d'apporter un regain d'espace cache puisque les niveaux L2 et L1 qui leurs sont propres et non-partageables se cumulent. De plus, la pression sur les caches de l'unique cœur auparavant utilisé est alors moins forte.

Tableau 4.11 Facteur d'accélération à l'issue de la phase 5 par rapport à MC^3 initial

	phase 1	phase 2	phase 3	phase 4	phase 5	phase 6
Fluide	4x	8x	8x	14x	**24x**	-
Élec.	17x	17x	17x	19x	**22x**	-
Mag.	13x	18x	18x	20x	**27x**	-
Rad. P1	14x	47x	47x	49x	**117x**	-
Rad. FVM	60x	237x	237x	239x	**357x**	-
Maillage	5x	12x	36x	39x	**78x**	-

Phase 6. Optimisation de l'efficacité mémoire

L'amélioration de l'efficacité mémoire pour les applications parallèles est une phase importante pour l'accélération puisqu'elle maximise l'efficacité globale du programme. L'impact est important pour l'efficacité de la vectorisation et permet de se rapprocher du facteur d'accélération théorique maximal.

Modification 6.a : Réutilisation des données en caches

Dans MC^3, à chaque appel du solveur de Helmholtz les relations de voisinage et les coefficients géométriques sont recalculés à la volée. Ces coefficients géométriques correspondent aux calculs des intégrales qui sont coûteuses à évaluer. Lorsque le maillage n'a pas été adapté et que le maillage est immobile (généralement dès que le disjoncteur est à pleine ouverture), ces coefficients deviennent constants. On peut donc les réutiliser d'une itération sur l'autre. De plus, grâce à l'isolation par module (par physique), on peut aussi proposer de mettre en commun les calculs liés aux intégrales de surface avec les calculs liés aux intégrales de volumes. Le partage de ces coefficients entre les physiques n'est pas si évident puisque les régions de calculs sont très différentes. Les améliorations de performance sont alors liées aux économies des points de vue arithmétique, du débit d'instructions et du chargement redondant des mêmes données.

On conserve les coefficients géométriques en mémoire et procède à leurs mises à jour locales lorsque le maillage a été adapté ou déplacé localement. En conjonction avec la proposition concernant la détection automatique des régions de calcul, on sait que les besoins en mémoire sont faibles. Les données auront toutes les chances de rester présentes dans les caches sans devoir être rechargées depuis la mémoire centrale. Particulièrement si la même physique est toujours planifiée et exécutée sur les mêmes processeurs/cœurs à chaque appel (*"thread affinity"*).

Modification 6.b : Renumérotation du maillage

La numérotation du maillage à un impact global et important sur les performances. L'impact de la renumérotation du maillage est global et important pour l'amélioration de la vitesse d'exécution. En effet, il participe à la proximité en mémoire des données (voir Figure 4.28) ce qui réduit le nombre requis d'accès à la mémoire principale car les données, si peu espacées, sont déjà dans les caches CPU. De plus, elle ne seront chargées en cache qu'une seule fois et réutilisées plusieurs fois avant d'être évincées. Sur la base des volumes finis, les relations de voisinage sont donc d'une grande importance, les voisins et leurs données associées doivent donc être stockés proches en mémoires pour maximiser la vitesse d'exécution. L'algorithme de renumérotation **R**everse-**C**uthill-**M**cKee (**RCM**) est un des plus efficaces mais séquentiel

par nature. Cependant, un simple tri par coordonnées croissantes est très facile à paralléliser et déjà très efficace, notamment car le ratio géométrique longueur/hauteur est supérieur à 8 pour les disjoncteurs ; c'est ce que nous avons mis en place. Sur la figure 4.28, l'exemple donné se base sur des volumes de contrôle de type barycentrique. Pour les volumes de contrôle triangulaires, le principe est le même : les voisins bords à bords doivent être le plus rapprochés que possible.

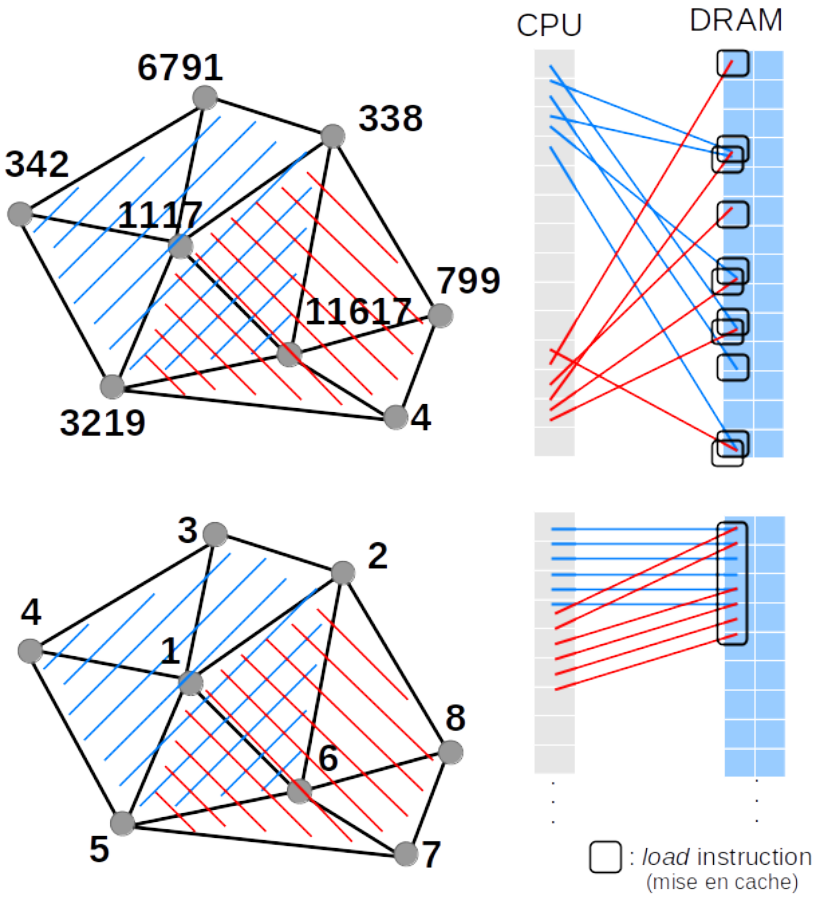


Figure 4.28 Impact de la numérotation du maillage sur la mémoire

Modification 6.c : Contrôle de l'alignement des données en mémoire

Le contrôle explicite de l'alignement des adresses des tableaux et des variables avec les lignes de caches maximise l'efficacité des accès mémoire et réduit le nombre d'accès à la DRAM et le nombre de lignes de caches gaspillées. L'alignement (en bytes) est spécifié au compilateur à l'aide de *pragmas* du type "*___attribute___((aligned(XXX)))*" ou lors de l'appel des allocateurs à la création des tableaux.

Modification 6.d : Architecture mémoire non-uniforme (NUMA)

En prenant en considération l'architecture matériel, on peut bénéficier d'accélérations supplémentaires. Les systèmes multi-sockets à base NUMA, généralement constitués de 2 à 8 CPU par nœud, partagent l'espace d'adressage mémoire entre les différents CPU pour rendre disponible à l'utilisateur un espace unique (d'une taille égale à la somme des espaces privés de chaque CPU). De plus, chaque CPU dispose de ses contrôleurs mémoire avec les limitations de débit et de latence qui leurs sont propres. Par exemple, un système à 4 CPU, permet donc de disposer de quatre fois la bande passante mémoire, quatre fois la quantité de cache, etc. Cependant, pour en tirer bénéfice les algorithmes doivent faire en sorte que chaque CPU accède à des données *locales* à ses contrôleurs mémoire. Une décomposition de domaine judicieuse et l'utilisation de la politique "*first-touch*" permettent d'initialiser les tableaux partiels dans des espaces mémoires spécifiques à chaque CPU. Utiliser plus de socket mais à nombre de threads constant permet d'accélérer encore la vitesse d'exécution. Les conséquence de cette proposition seront étudiées au chapitre 5, elles ont un impact sur la vitesse d'exécution et la scalabilité du programme.

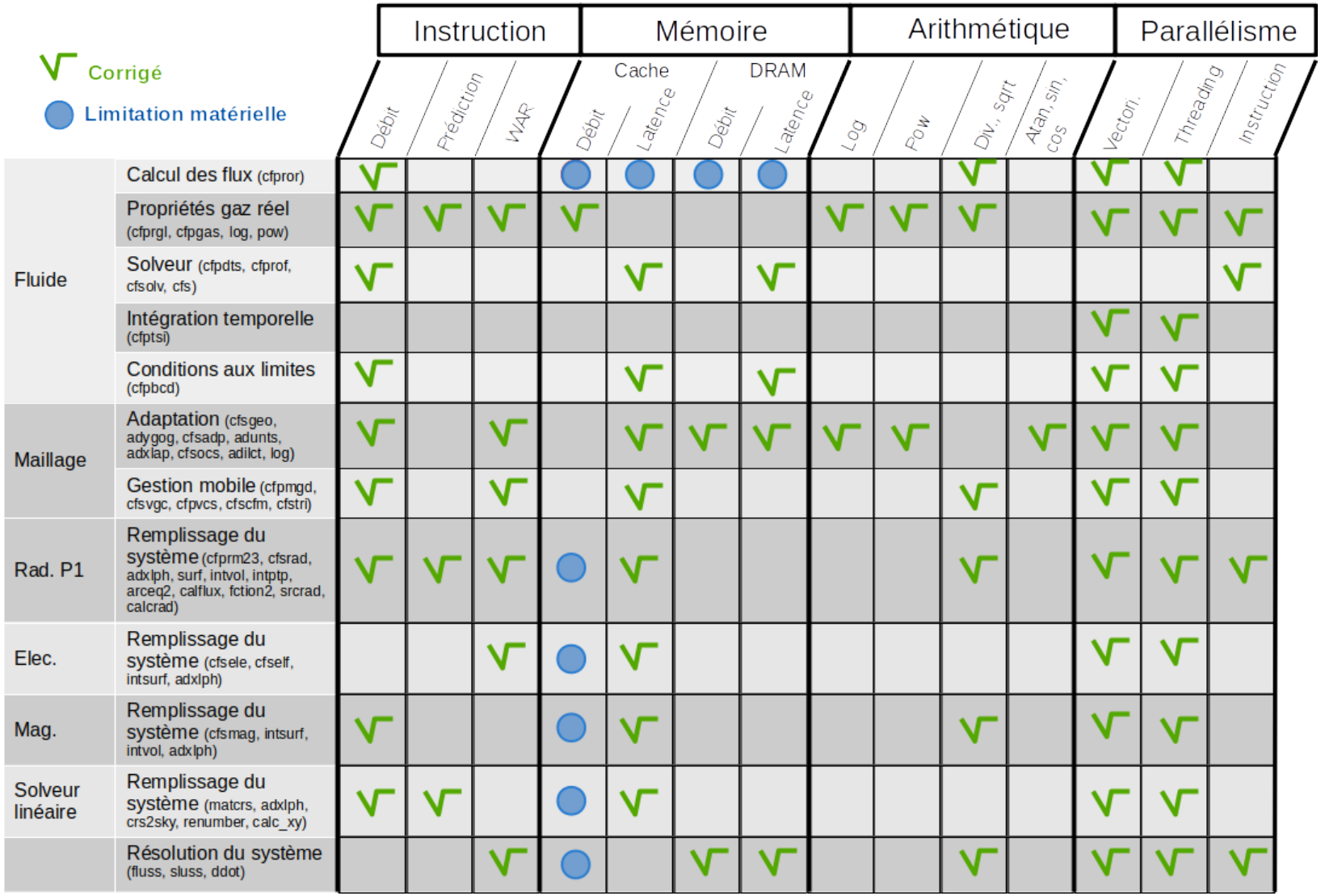
Résultats de performance de la phase 6 :

Une bonne partie des algorithmes sont limités en vitesse d'exécution par la latence et le débit mémoire. La vectorisation est très sensible à l'alignement des adresses DRAM avec les lignes de caches. Ils s'agissait donc d'une composante importante pour maximiser l'efficacité de la vectorisation et de la parallélisation et finalement du programme dans son ensemble aussi. Les phases précédentes ont pris en compte cela, notamment dans les choix d'algorithmes à accès séquentiels unitaires en mémoire et par des structures de données *"simples"*. Le tableau 4.12 présente les accélérations de la dernière phase de modernisation et le tableau 4.13 résume l'impact sur les limitations en performance.

Tableau 4.12 Facteur d'accélération à l'issue de la phase 6 par rapport à MC^3 initial

	phase 1	phase 2	phase 3	phase 4	phase 5	phase 6
Fluide	4x	8x	8x	14x	24x	**32x**
Élec.	17x	17x	17x	19x	22x	**37x**
Mag.	13x	18x	18x	20x	27x	**33x**
Rad. P1	14x	47x	47x	49x	117x	**161x**
Rad. FVM	60x	237x	237x	239x	357x	**600x**
Maillage	5x	12x	36x	39x	78x	**125x**

Tableau 4.13 Améliorations des facteurs limitant les performances de MC^3



4.5 Résumé des validations numériques réalisées

Pour s'assurer de l'exactitude des résultats de nombreux cas test ont été réalisés, les plus importants et les résultats associés sont présentés dans cette section. Ils ont été utiles pour apprécier la précision vis-à-vis de solutions analytiques et ont été également utilisés en temps que tests de non-régression lors des étapes incrémentales de modification (et d'accélération) du code. Mis à part le cas test présenté à la section 4.5.2, pour la résolution du potentiel électrique, les tests de validation ont démontrés une bonne concordance avec la version de référence.

4.5.1 Fluide (maillage statique)

Pour faciliter la validation numérique, dans les cas tests qui suivent, une table avec des données thermodynamiques correspondantes à un gaz parfait a été utilisée.
Tube à choc :

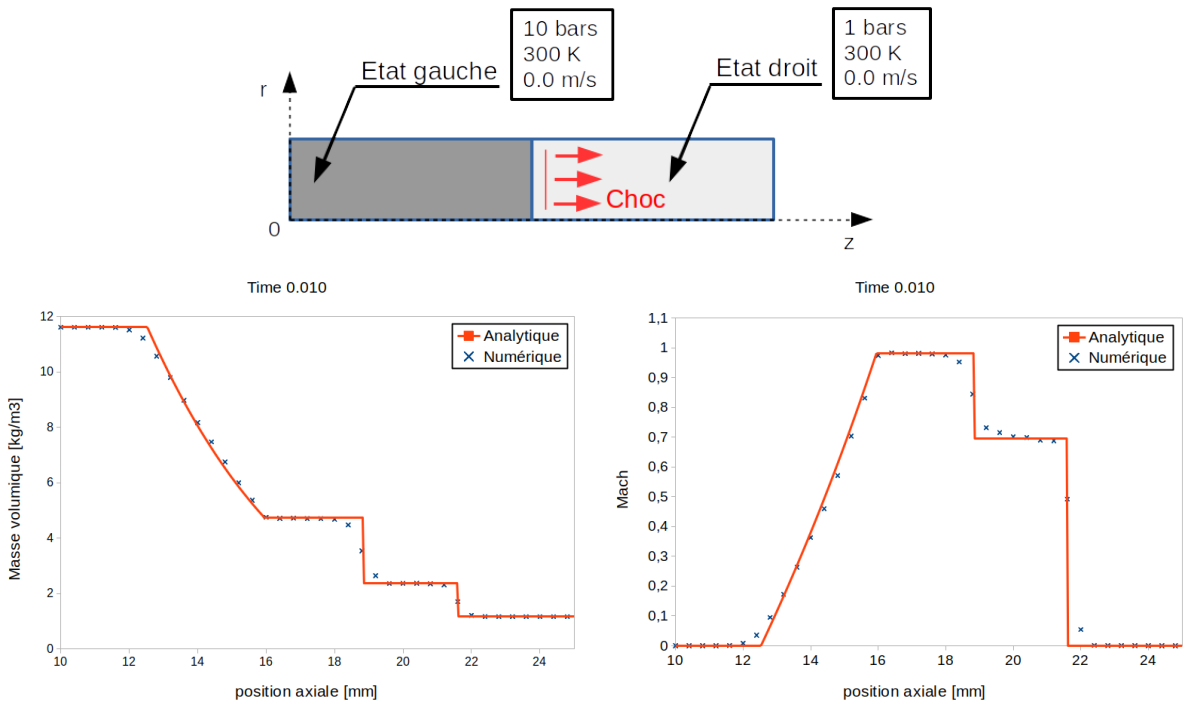


Figure 4.29 Cas de validation : tube à choc

Les résultats sont valides vis-à-vis des solutions analytiques ([120]) et équivalents à la version initiale de MC^3 (Figure 4.29), et pour tous les autres temps de simulation (non présentés ici).

Écoulement radial :

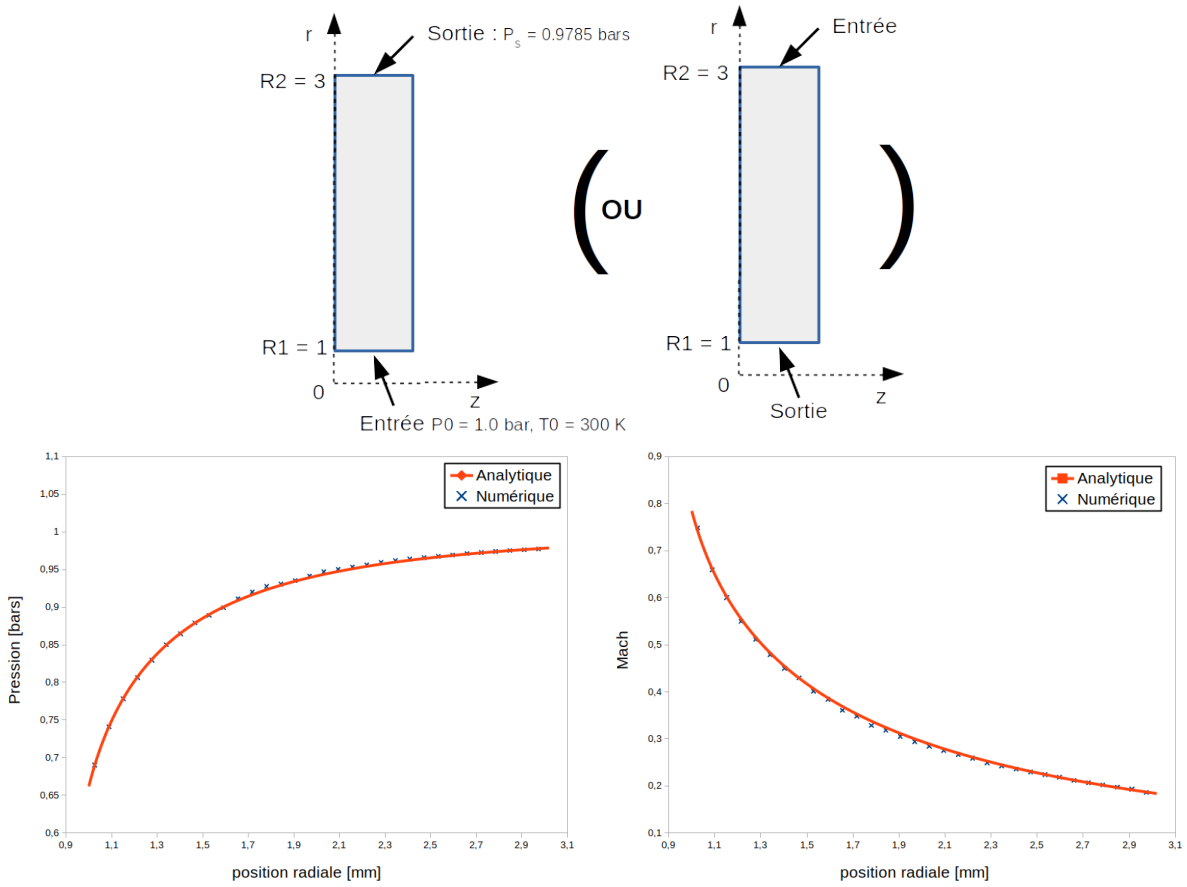


Figure 4.30 Cas de validation : écoulement radial stationnaire (équivalence avec une tuyère divergente : *à gauche* ou convergente : *à droite*

Pour l'écoulement radial (Figure 4.30), les résultats sont également valides vis-à-vis de la solution analytique ([5]) et comparable avec la version de référence de MC^3.

4.5.2 Potentiel et champ électrique

Doubles sphères concentriques :

La validation des calculs du potentiel et du champ électrique a été réalisé avec le cas test présenté à la figure 4.31, et met en évidence les points faibles de MC^3 (Figure 4.32) pour ce qui est des valeurs calculées sur l'axe de symétrie. En effet, l'erreur y est importante et dissymétrique par rapport à la sphère intérieure. Ceci est dû à un maillage généré par MC^3 qui n'est pas symétrique et qui induit des calculs de gradients qui comportent trop peu d'informations directionnelles. En revanche, pour les calculs effectués avec la nouvelle méthode cartésienne (Figure 4.33), pour des tailles caractéristiques des côtés des triangles équivalentes au calcul $MC3$, les résultats sont nettement améliorés sur le point de vue de la précision vis-à-vis des solutions analytiques ([94, 95]) et de la symétrie. Les valeurs de champ électrique sur l'axe de symétrie sont un critère très important pour la conception des disjoncteurs, notamment au niveau de la tige, cela constitue une excellente amélioration du point de vue de l'industriel.

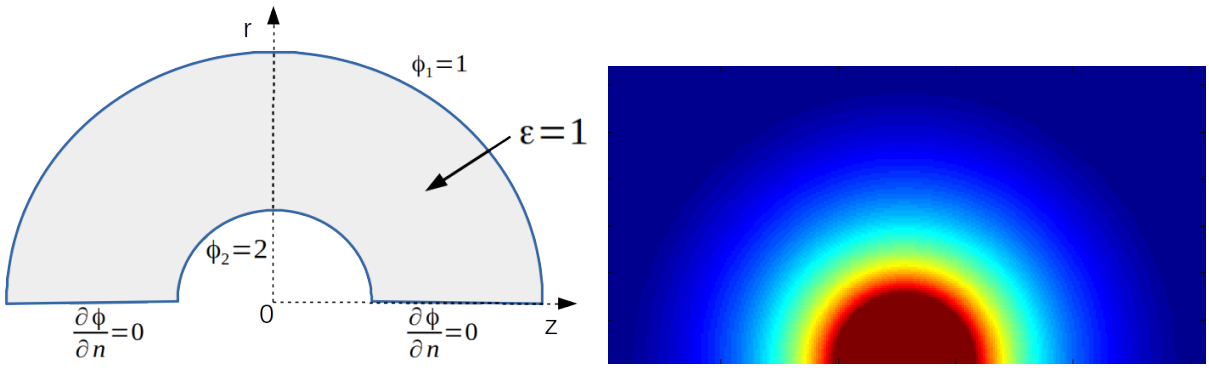


Figure 4.31 Cas de validation : doubles sphères concentriques : configuration, potentiel électrique calculé

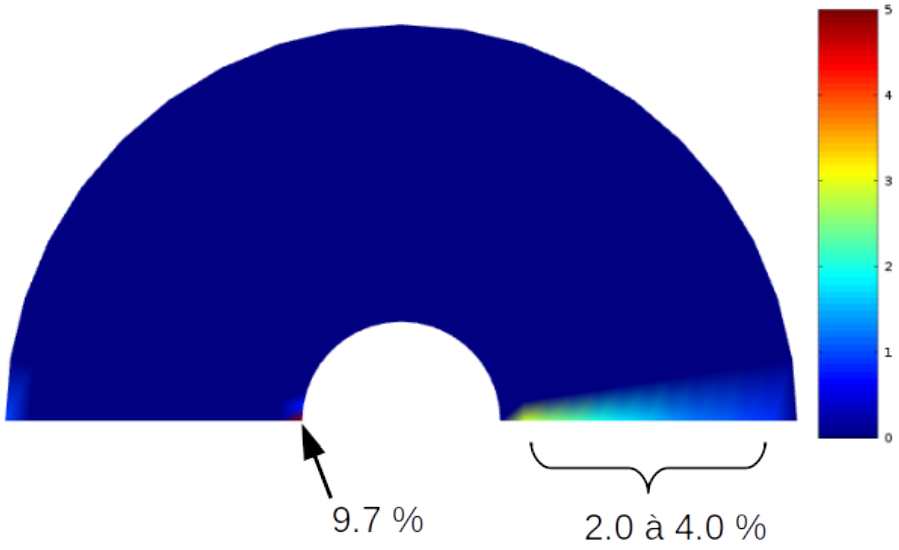


Figure 4.32 Cas de validation : doubles sphères concentriques : erreur sur le champ électrique pour MC^3

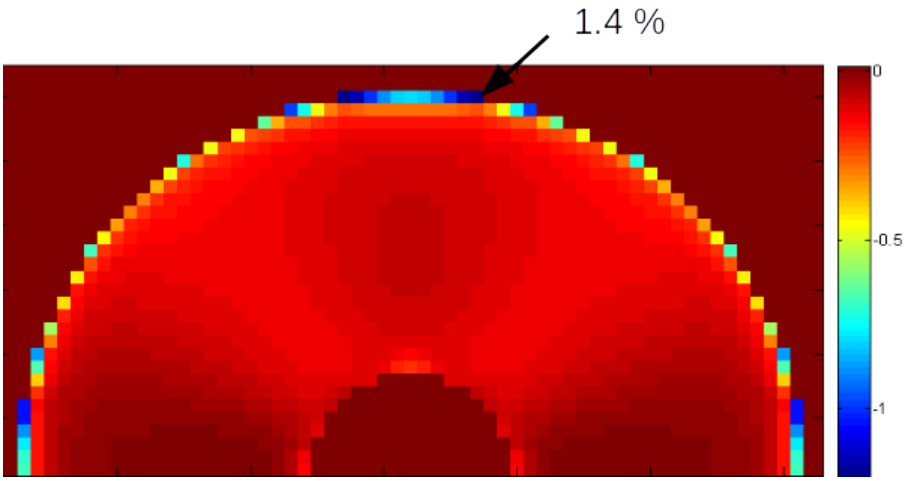


Figure 4.33 Cas de validation : doubles sphères concentriques : erreur sur le champ électrique pour maillage cartésien

4.5.3 Champ magnétique

Doubles cylindres concentriques :

D'après les résultats qui suivent (Figure 4.34), les calculs du champ magnétique sont valides vis-à-vis des solutions analytiques ([94, 95]) et similaires à la version de référence de MC^3 et aucun phénomène non-physique n'apparaît aux frontières de la géométrie.

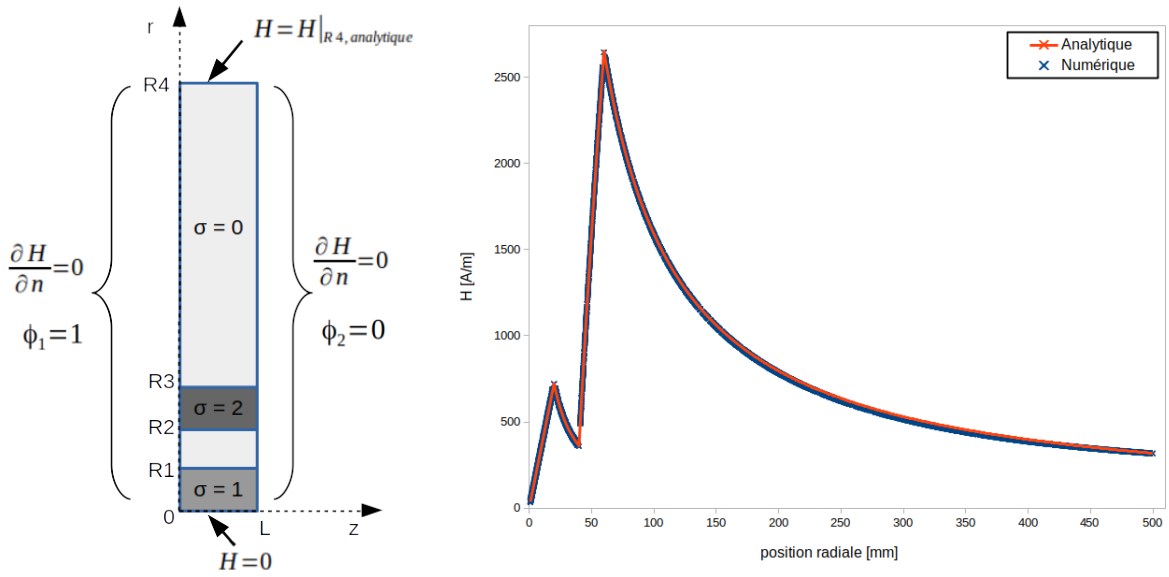


Figure 4.34 Cas de validation : doubles cylindres concentriques : champ magnétique

4.5.4 Rayonnement thermique

Cylindre avec média participant :

Les résultats des deux modèles de rayonnement, présentés à la figure 4.35, sont semblables à ceux de Melot [76] et sont valides selon les solutions analytiques ([24, 58]) concernant le modèle FVM. En revanche, pour le modèle P1, la dépendance axiale du flux n'est pas prise en compte, plus le coefficient d'absorption est élevé et plus les valeurs sont surestimées. Il s'agit d'un comportement normal du modèle P1, lié au type d'équations résolues.

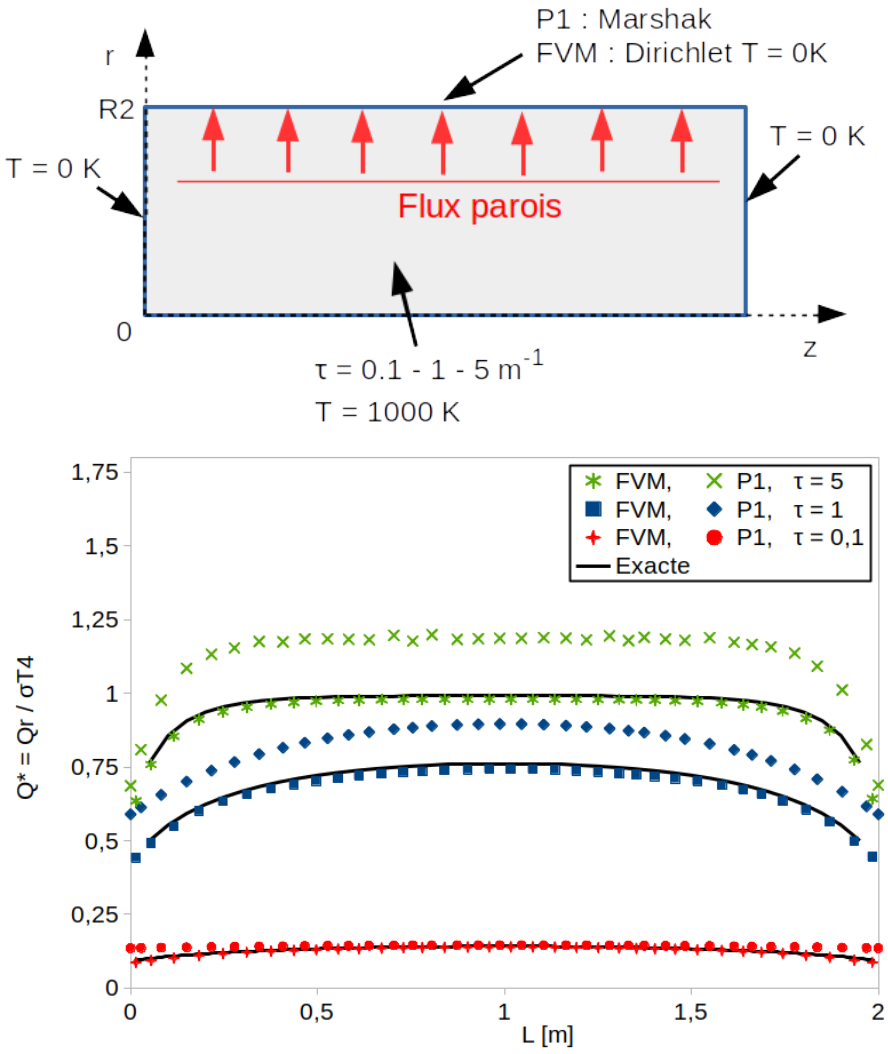


Figure 4.35 Cas de validation : cylindre avec média participant : flux à la paroi supérieure

4.5.5 Maillage mobile

Mouvement aléatoire des mailles intérieures :

Dans ce cas test, on initialise le domaine avec un champ de vitesse nul et des valeurs constantes pour les autres propriétés. Ensuite, les côtés des triangles sont déplacés aléatoirement dans l'espace au fil des itérations (figure 4.36). La conservation des propriétés fluides doit être assurée lorsque les points A, B et C sont déplacés aléatoirement et que le triangle ABC est déformé.

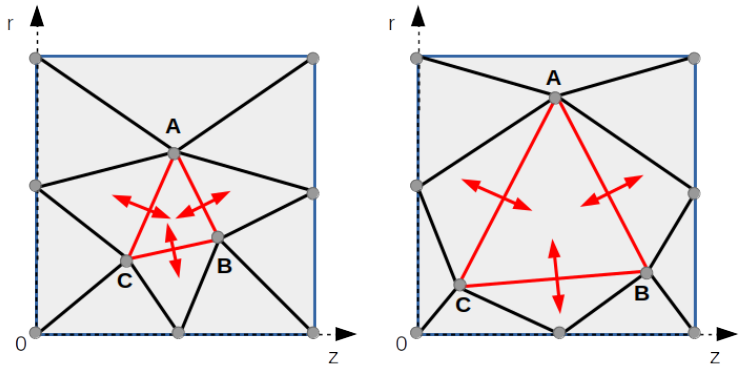


Figure 4.36 Cas de validation : mouvement aléatoire des mailles intérieures

L'erreur sur la masse volumique est inférieure à 10^{-8} $kg.m^{-3}$ après quelques milliers d'itérations et des agrandissements et réductions aléatoires successives du triangle ABC d'approximativement ±150%. Les lois de conservations géométriques (GCL) sont donc vérifiées.

Piston de compression :

Le déplacement brusque d'une des parois génère une onde de compression qui se propage dans le domaine (Figure 4.37). Les résultats concordent avec la solution analytique. Ces premiers cas tests valident l'aspect mobile du maillage et du solveur fluide ALE.

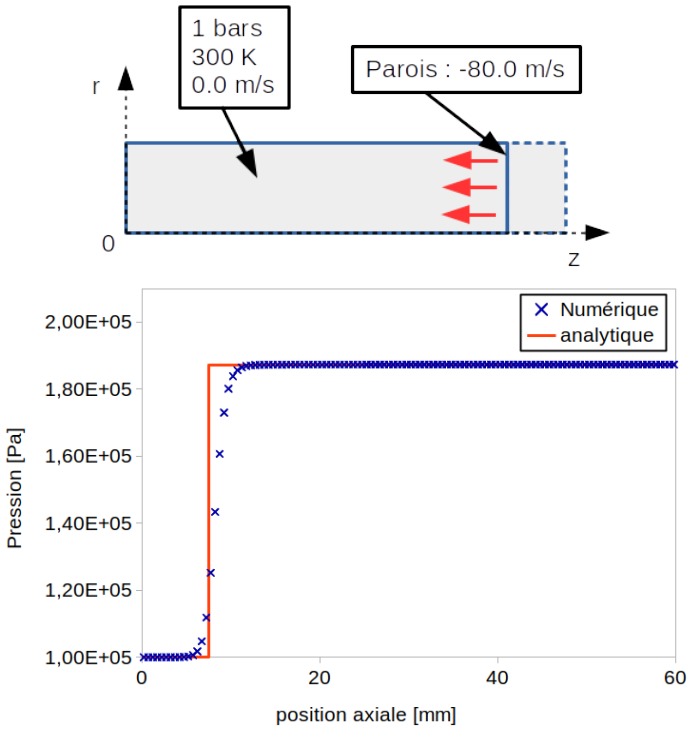


Figure 4.37 Cas de validation : piston de compression

4.6 Résumé de la phase d'accélération

La méthodologie globale d'accélération que nous avons adoptée comporte les tâches suivantes :

- l'analyse de l'existant : profilage complet et détaillé et identification des portions à améliorer,
- la compréhension détaillée du code,
- la réécriture du code et/ou des structures de données en prenant en compte les spécificités de l'application,
- la validation des gains.

Facteurs d'accélération possibles :

- de l'ordre de 1.5x-2x par des actions mineures de nettoyage et réorganisations mineures,
- de l'ordre de 30x à 600x suivant les physiques pour les réécritures d'algorithmes et structures de données et l'utilisation plus efficace de la vectorisation/parallélisation.

Remarque : le modèle de rayonnement FVM, était inutilisable au niveau industriel pour des raisons de lenteur en temps d'exécution. Avec les optimisations réalisées, il est maintenant envisageable de l'utiliser en industrie, car le temps d'exécution par appel est désormais inférieur à une seconde. Du point de vue industriel, une calibration des conditions aux limites sera nécessaire.

A ce stade, les physiques ont été accélérées indépendamment, la prochaine étape consiste à connecter leurs nouvelles implémentations réalisées en C++. Pour réaliser ces connexions multiphysiques, une nouvelle architecture logicielle doit être conçue. Les calculs des termes sources d'arc, provenant des différentes physiques, doivent notamment être synchronisés avec les pas de temps et les calculs de mécanique de fluides. La conception de la nouvelle architecture adaptée au contexte multiphysiques est décrite dans le prochain chapitre (5).

CHAPITRE 5 ARCHITECTURE LOGICIELLE ADAPTÉE AU CONTEXTE MULTIPHYSIQUES

5.1 Introduction à l'architecture logicielle

La conception d'une architecture logicielle est une tâche cruciale lors de la création d'un logiciel. En effet, une fois réalisée, elle ne changera pas au cours de la vie du logiciel. Lorsqu'elle doit évoluer, un nouveau logiciel est créé et les caractéristiques de l'architecture sont redéfinies. Les performances et la qualité finales d'une application sont dépendantes de son architecture de logiciel qui doit donc être réalisée avec soin.

L'architecture d'un logiciel décrit l'agencement des différents éléments qui le composent et qui lui permettent de fonctionner. La collection de composants, leurs relations et leurs interactions sont généralement définies à l'aide de diagrammes descriptifs. L'architecture représente un compromis entre les exigences du client et les coûts de conception. Les performances finales du logiciel, sont un des nombreux points à prendre en compte, l'application devra aussi être facile à prendre en main, suffisamment souple et évolutive pour faciliter les mises à jour futures. La fiabilité et la compatibilité avec les outils et matériels existants sont d'autres points à considérer. D'une description de bonne qualité de l'architecture, il sera possible d'obtenir un logiciel de bonne qualité.

Afin de concevoir une bonne architecture, donc un bon logiciel, il est vital d'avoir une vision d'ensemble approfondie de toutes les fonctionnalités, modules, dépendances, et de *visualiser* conceptuellement le projet dans sa globalité. Lorsqu'il s'agit de la modernisation d'un logiciel existant, il faut alors s'approprier ses algorithmes, son code source, ses modèles, s'imprégner complètement des grandes lignes, aux détails subtils, ce qui permettra de concevoir une architecture pertinente et performante. Le travail synthétisé dans le chapitre 4 sur l'analyse de MC^3 et l'expérience passée sur le sujet des disjoncteurs haute-tension nous l'offrent. Le regard porté ne doit pas s'orienter seulement d'un point de vue informatique. Dans le chapitre précédent, on a vu que la plupart des opportunités d'accélération provenaient plutôt d'une connaissance approfondie des méthodes numériques, des physiques entrant en jeu et des particularités du champ d'application des disjoncteurs. La partie informatique correspond seulement à la programmation dans les règles de l'art et la parallélisation des tâches qui en découlent, avec des techniques de programmation avancées qui, seules, ne suffisaient donc pas pour accélérer MC^3. Ce qui justifiait cette partie du travail doctoral. Les modules de chaque physique ont été optimisés et dans ce chapitre, ils vont être couplés dans un solveur adapté aux résolutions multi-physiques.

5.2 L'architecture de MC^3, dites-vous ?

MC^3 ne possède pas d'architecture à proprement parler, c'est un code spaghetti dont le déroulement des appels de fonctions est difficile à suivre. Au fil des nouveaux développements, réalisés pour la plupart par des étudiants de l'école Polytechnique de Montréal, des ajouts et extensions du programme ont été intégrés. Le code source est aujourd'hui morcelé et difficilement maintenable et la prise en main par un nouveau développeur requiert plusieurs mois de lourds investissements. De plus, la documentation est trop basique et inexistante pour certaines portions du programme. En conséquence, l'extension de MC^3 avec de nouveaux modèles qui peuvent être 3D, est jugée trop coûteuse en efforts de développement, et la situation critique du code source serait encore accentuée. A titre d'exemple, la figure 5.1 illustre graphiquement les appelants de la fonction de mise à jour des propriétés primitives de gaz réel $cfprgl$ dans la version non optimisée de MC^3. On remarque que de nombreux chemins sont possibles pour appeler cette seule fonction, la situation est encore plus complexe avec l'ensemble des fonctions du programme (non lisible graphiquement à l'échelle de ce document, mais représenté en aperçu à la figure 5.5).

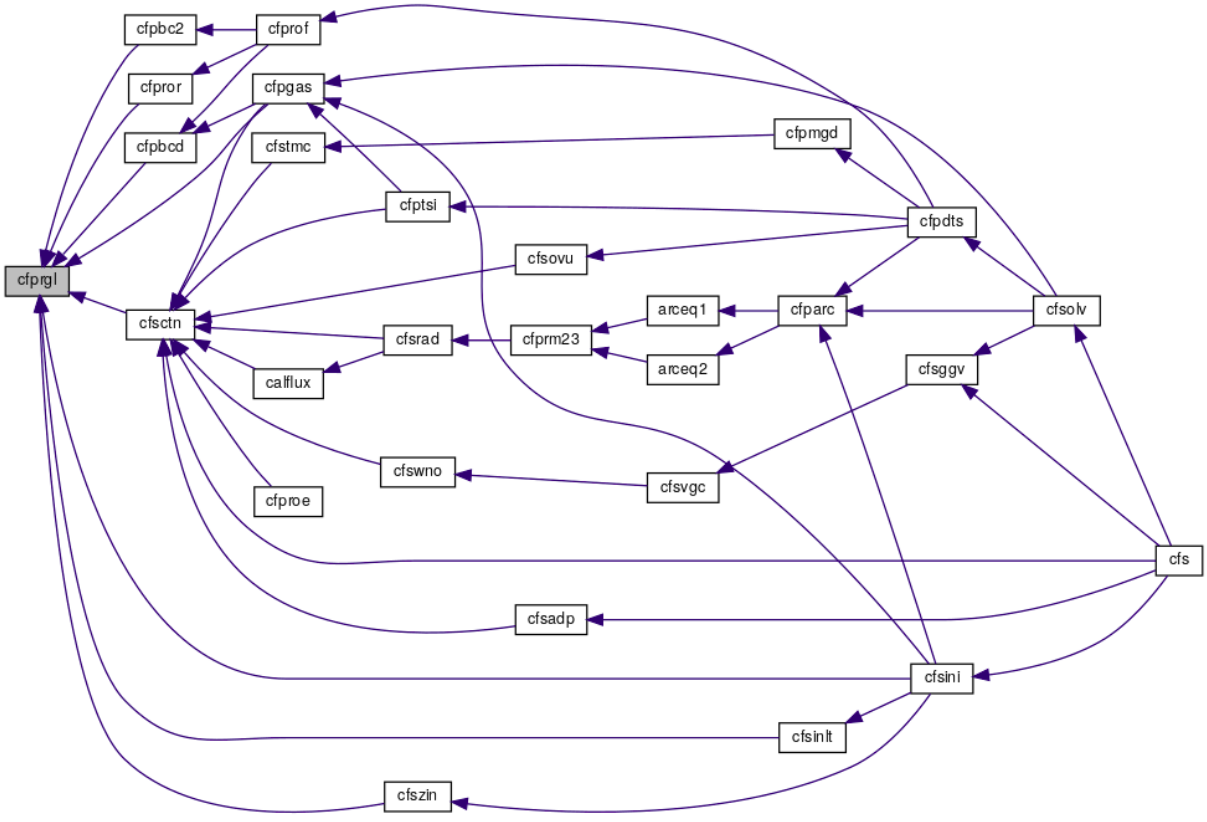


Figure 5.1 Graphe des appelants de $cfprgl$

5.3 Nouvelle architecture

Dans la section 4, les différents modules ont été isolés et optimisés incrémentalement en conservant la structure globale du programme. Les différents modèles physiques doivent être couplés de façon adaptée et exposer le parallélisme de tâches. Les opérations de maintenance représentent le facteur le plus important dans le coût financier global du logiciel, en effet elles y contribuent avec une portion supérieure à 95 % (Pigoski [93]). La performance en temps de calcul, la lisibilité du code source et la facilitation des évolutions futures et de la maintenance sont des points qui peuvent se révéler contradictoires. L'architecture doit être conçue pour présenter un équilibre optimal entre ces contraintes. D'un point de vue conceptuel plus global, l'amélioration du logiciel de simulation des chambres de coupure des disjoncteurs haute-tension MC^3 doit prendre en compte tous les aspects, liés d'une part au temps requis pour toutes les étapes d'un calcul, et d'autre part au temps requis pour la maintenance de l'outil scientifique.

Plus particulièrement, ces aspects concernent :

1. Le temps nécessaire à la maintenance du logiciel (débogage, validation).
2. Le temps passé dans les nouveaux développements et leurs intégrations dans l'outil.
3. Le temps requis à la génération de la documentation développeur et des mises à jour.
4. Le temps de création de la documentation utilisateur et des mises à jour.
5. Le temps dépensé pour les tâches de pré-traitement (géométrie, C.L., paramètres de calculs, etc).
6. Le temps machine d'exécution des calculs.
7. Le temps de gestion et de suivi des calculs par les utilisateurs.
8. Le temps indispensable à l'analyse et plus particulièrement aux tâches requises lors du post-traitement (extraction de données, tracés, etc).

L'architecture concerne directement les points 1 à 4 et 6. Les points 5, 7 et 8 peuvent également être améliorés par la conception du programme et les choix techniques réalisés. Comme premier exemple réalisé dans ce projet : l'abandon du format natif SOL des solutions MC^3 vers un format standard, public, maintenu activement, très optimisé et largement diffusé tel que VTK, permet de s'affranchir du développement, de la maintenance et de la génération de plugins de lecture pour les outils de visualisation et de post-traitement tels que Visit ou Paraview. En second exemple, le passage au langage de programmation C++, combiné à un environnement de développement moderne et efficace, permet d'améliorer principalement les

points 1 à 4 et 6 d'un facteur important. Les points 5, 7 et 8 sont indirectement améliorés en autorisant l'utilisation de librairies graphiques, de librairies de calculs parallèles et de nouveaux outils modernes plus intuitifs pour les utilisateurs. Les futurs développeurs auront la responsabilité de la documentation du code source, de la validation et des tests automatiques. La mise en place de l'outil Doxygen pour la gestion de la documentation et de l'outil GIT pour la gestion des versions du code source permettront d'améliorer la qualité logicielle sur les points 1 à 4.

Les critères de design qui ont guidés l'élaboration de la nouvelle architecture s'appuient sur différents ouvrages de référence, apportant une culture sur l'architecture logicielle. Hofmeister et al. [42] proposent un ouvrage très complet sur les bonnes pratiques et les techniques pour produire des designs architecturaux de qualité. Ils proposent des concepts pour déterminer les priorités dans les besoins et quelques clefs pour aboutir avec succès à une solution architecturale. Bosch [14] avance une approche plus industrielle dans le design d'architectures logicielles. Albin [4] traite le sujet dans son ensemble et aborde avec précision le thème des cadres de travail pour l'architecture logicielle. Bass et al. [8] donnent une vision plus appliquée et plus moderne des bonnes pratiques et des concepts architecturaux, avec une description des cycles de vie. Leur philosophie se base profondément sur des critères de qualité. Buschmann et al. [17] proposent des modèles de conception à différents niveaux d'abstraction afin d'orienter la conception d'architecture logicielle. Les modèles se basent sur l'expérience et se combinent pour développer tout un système de modèles multi-niveaux. Clements et al. [23] se focalisent plutôt sur l'aspect de la documentation de l'architecture. En effet, la conception architecturale est une étape de la création, sa description permet qu'elle soit comprise et communiquée avec succès aux différents protagonistes du projet.

Afin de décrire notre architecture, on propose d'utiliser les diagrammes de la norme UML [84]. Dans cette thèse, les diagrammes de cas d'utilisation (Figure 5.2), d'activité (Figure 5.3) et de classes (Figure 5.4) seront présentés. Ils représentent les principaux diagrammes d'intérêts pour la description architecturale.

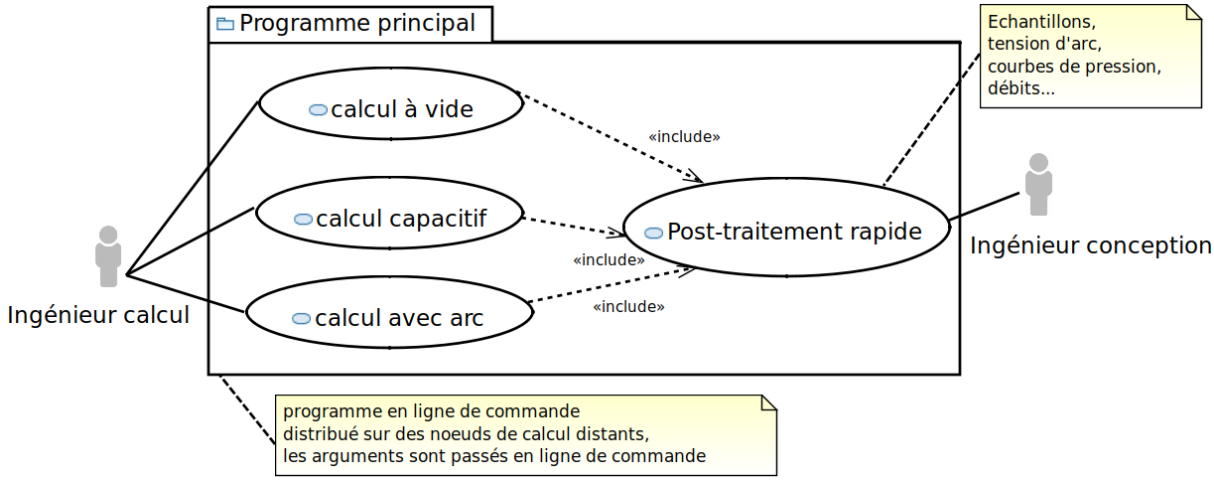


Figure 5.2 Cas d'utilisation

On distingue trois types d'utilisation distincts suivant le type de calcul, certaines physiques seront alors actives. Le diagramme d'activité pour le programme principal est présenté à la figure 5.3. Son rôle est de récupérer les options transmises en arguments et de créer/lancer le solveur multi-physiques ("Créer/Lancer solveur"), contrairement au cas de MC^3 qui remplissait une partie des rôles du solveur. Les actions conditionnelles dépendent du type de calcul sélectionné et/ou de l'instant dans le calcul. Par exemple, avant la séparation des contacts électriques seul le calcul de mécanique des fluides est réalisé, puis lorsque l'arc apparaît les autres physiques sont à leur tour résolues. Le diagramme de classes pour le paquet du solveur multi-physiques est présenté à la figure 5.4. Les classes sont simplifiées pour améliorer la visibilité de la figure. Chacune d'entre elles est indépendante. Ainsi pour écrire un solveur personnalisé, un développeur peut intégrer seulement les physiques pour lesquelles il est intéressé, aux dépendances près, telles que le maillage par exemple ou ajouter de nouveaux modules. Le diagramme de paquetages est très similaire au diagramme de classes et n'est donc pas présenté dans ce document. En effet, le découpage du solveur et les groupes de classes correspondent à la représentation qui est faite sur le diagramme de classes.

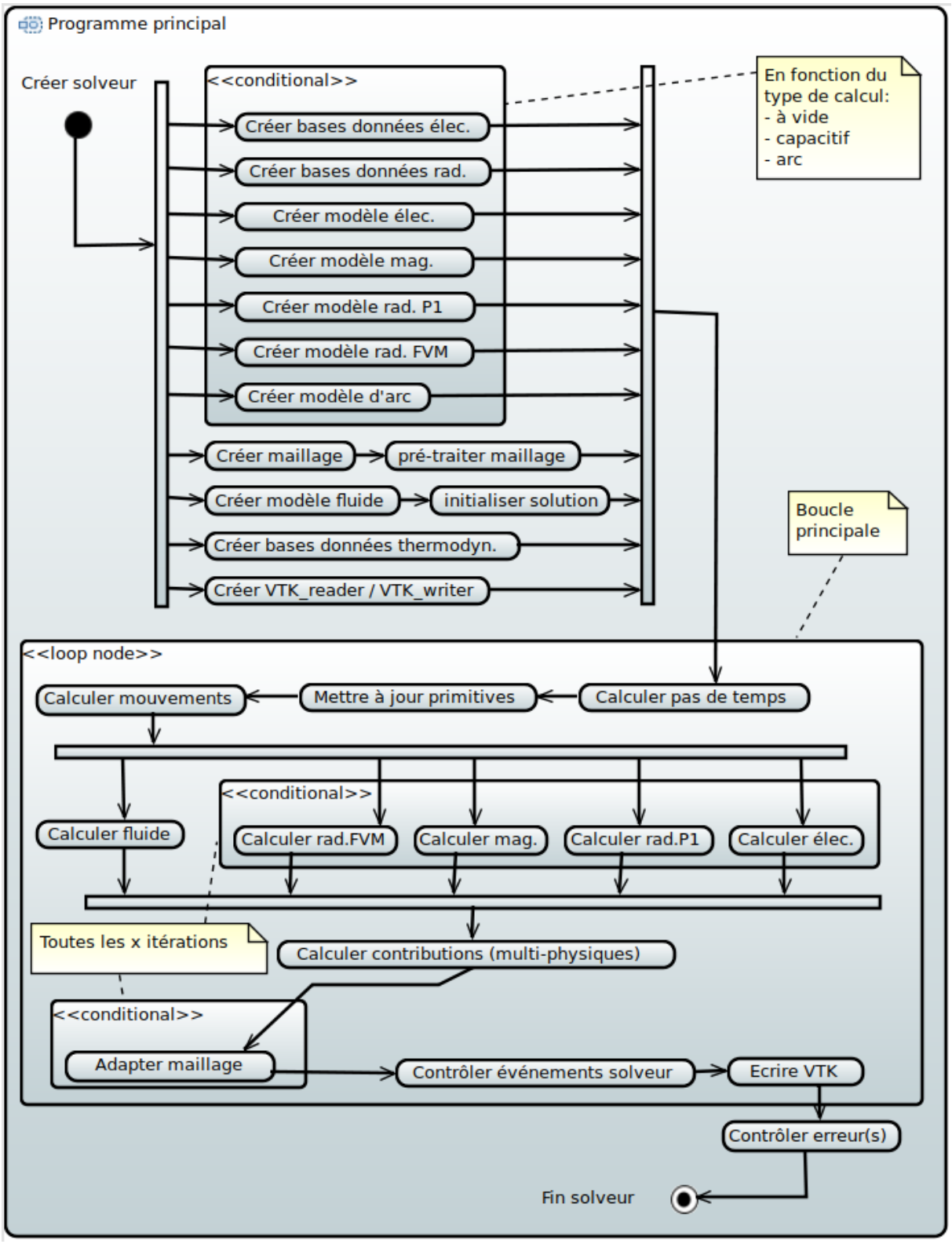


Figure 5.3 Diagramme d'activité

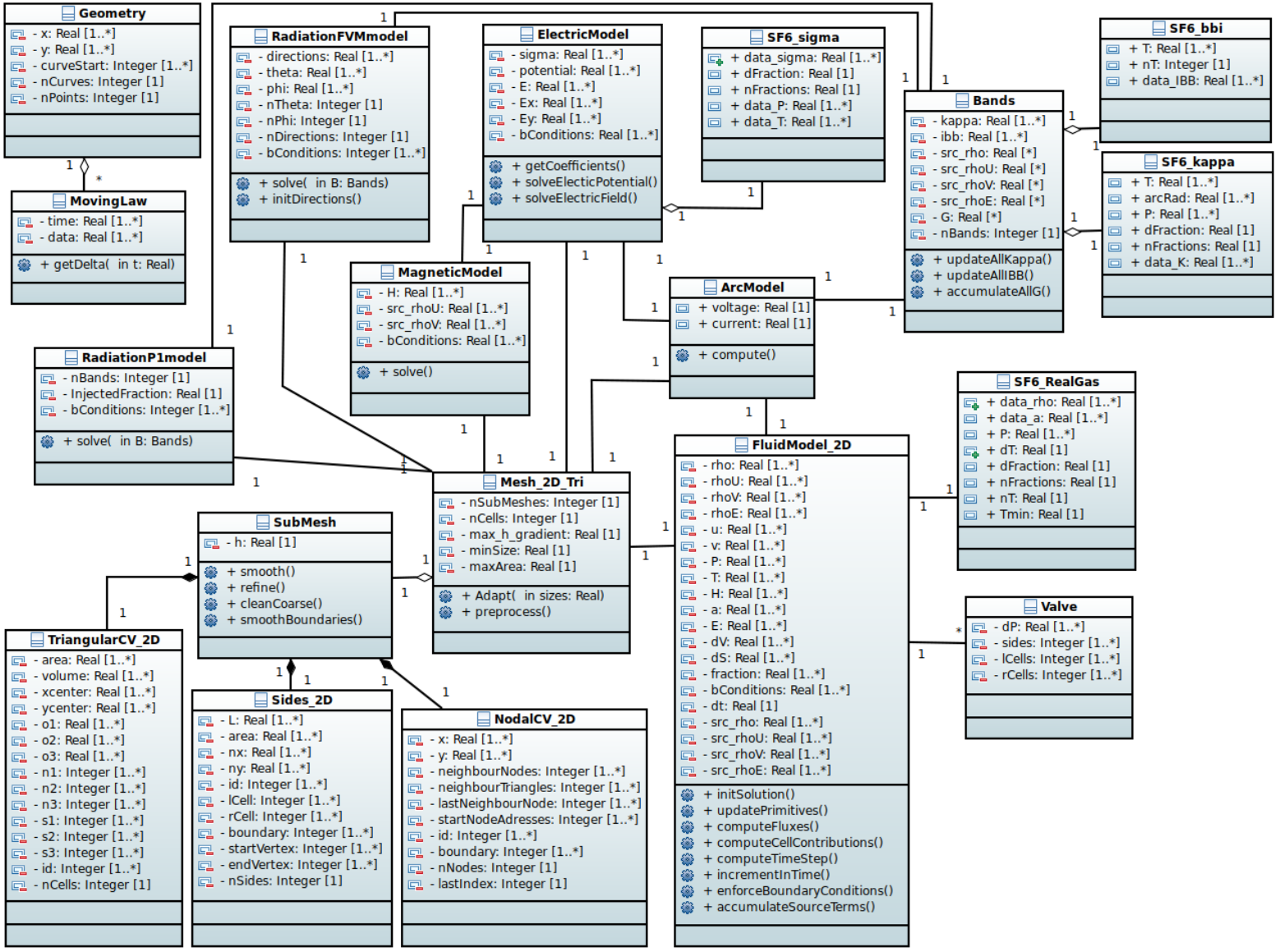


Figure 5.4 Diagramme de classes (simplifiées) pour le paquet du solveur multi-physiques

5.4 Est-ce la bonne architecture ?

Les acquis de l'expérience faite avec MC^3 ont été fortement mis à contribution dans la démarche de conception. La structure proposée permet un équilibre entre les compromis de performance en vitesse d'exécution, de facilité de maintenance et d'évolutivité de l'outil. La compréhension et la lisibilité du code source sont grandement améliorées par l'architecture proposée et le style d'implémentation. La combinaison avec la documentation intégrée Doxygen et le système de gestion de version GIT permet de réduire beaucoup les temps nécessaires à la prise en main, à la maintenance du code et aux tests unitaires de validation, tout en réduisant de manière importante les coûts financiers associés (points très majoritaires dans l'ancien outil MC^3).

Selon la norme ISO 25010 (exigences et évaluation de qualité des systèmes et du logiciel), tous les indicateurs de la qualité du logiciel sont améliorés :

- La capacité fonctionnelle, qui répond, selon un certain taux de couverture, aux exigences des clients (ingénieurs de conception, de calcul) et/ou à la spécification, a été améliorée. Dans MC^3 des fonctionnalités requises par les utilisateurs étaient absentes, comme par exemple la possiblité de réaliser des calculs avec deux arcs consécutifs (calcul OCO).
- La fiabilité a été fortement augmentée : les tests unitaires réussissent tous et sont importants car un défaut mineur peut générer des défaillances majeures. De plus, la tolérance aux erreurs utilisateurs et matériel est gérée, et plus importante que dans MC^3, sujet à de nombreuses erreurs de segmentation par exemple. L'architecture répond au besoin de décomposer indépendamment et facilement les modules pour réaliser les tests de validation.
- La performance est beaucoup plus importante qu'auparavant (section 4) et le rapport entre les ressources matérielles utilisées et la quantité de résultats délivrés sont largement augmentés.
- La facilité d'utilisation est améliorée : la compréhension, l'apprentissage et l'exploitation sont aisés. Lors des erreurs d'utilisation, l'utilisateur est guidé, ce qui n'entraîne pas d'erreur de dysfonctionnement.
- La maintenabilité et l'extensibilité sont aisées, tel que décrit ci-haut, la complexité du code est réduite et le respect des règles de codage et de conception est renforcé. Ces points étaient très critiques pour MC^3.
- La portabilité est conservée : malgré certaines parties critiques, écrites directement en assembleur ou à l'aide de fonctions de vectorisation intrinsèques, la portabilité

est assurée pour les futures générations de processeur à architecture X86_64 dont la rétro-compatibilité est continue de génération en génération (très pérenne).

— La confiance dans l'outil est améliorée grâce aux tests de régression et aux vérifications des données d'entrées utilisateurs (robustesse et tolérance aux erreurs utilisateurs).

L'interface graphique, qui accompagnera le solveur, renforcera encore plus ces indicateurs de qualité et ajoutera de nouvelles améliorations concernant les critères de confort et de plaisir d'utilisation (ISO 25010) et constituera des développements futurs chez l'industriel. Le patron de conception sera basé sur le Modèle-Vue-Contrôleur (MVC), dans lequel l'interface graphique sera indépendante de la partie contrôle et de la partie solveur (distribué sur un cluster de calcul). La nouvelle structure C++ sous forme de classes rend l'interconnexion multiphysiques naturelle dans le paquet du solveur multiphysiques (Figure 5.4) si bien qu'aucun patron de conception spécifique ne lui est réellement appliqué. Néanmoins, le paquet du solveur multiphysiques agit comme l'interface de communication entre les différents objets dynamiquement créés lors de l'exécution, et présente quelques points de similitudes avec le patron *Puppeteer Pattern*, analysé dans un contexte de modélisation multiphysiques par Rouson et al. [107] . De plus, la décomposition en terme de tâches exécutées en parallèle a été réalisée à différents niveaux d'imbrication. En effet, pendant la même itération, toutes les physiques ont été rendues indépendantes et les appels et les résolutions sont réalisés avec des tâches parallèles non-bloquantes imbriquées. Les résolutions ne nécessitent pas de synchronisation, excepté avant l'intégration temporelle et à la fin d'une itération. Le scheduler de la bibliothèque TBB se charge de mapper dynamiquement les tâches sur les threads selon les contraintes du matériel (proximité en cache, affinité au fil des itérations, etc). Ainsi, la vitesse d'exécution et la scalabilité sont élevées (Figure 5.10) même lorsque la taille du problème reste contenue et les performances continueront de s'améliorer avec les générations futures de CPU, sans efforts particulièrement importants.

Le passage d'un langage procédural (FORTRAN 77) à un langage orienté objet (C++) a aussi permis de mieux structurer le code source tel qu'illustré sur la figure 5.5 qui représente le graphe d'appel des fonctions.

Tous ces éléments ainsi que les performances qui sont au rendez-vous (section suivante), nous permettent d'affirmer que l'architecture semble bien adaptée, et seules les futures années d'exploitation pourront le confirmer.

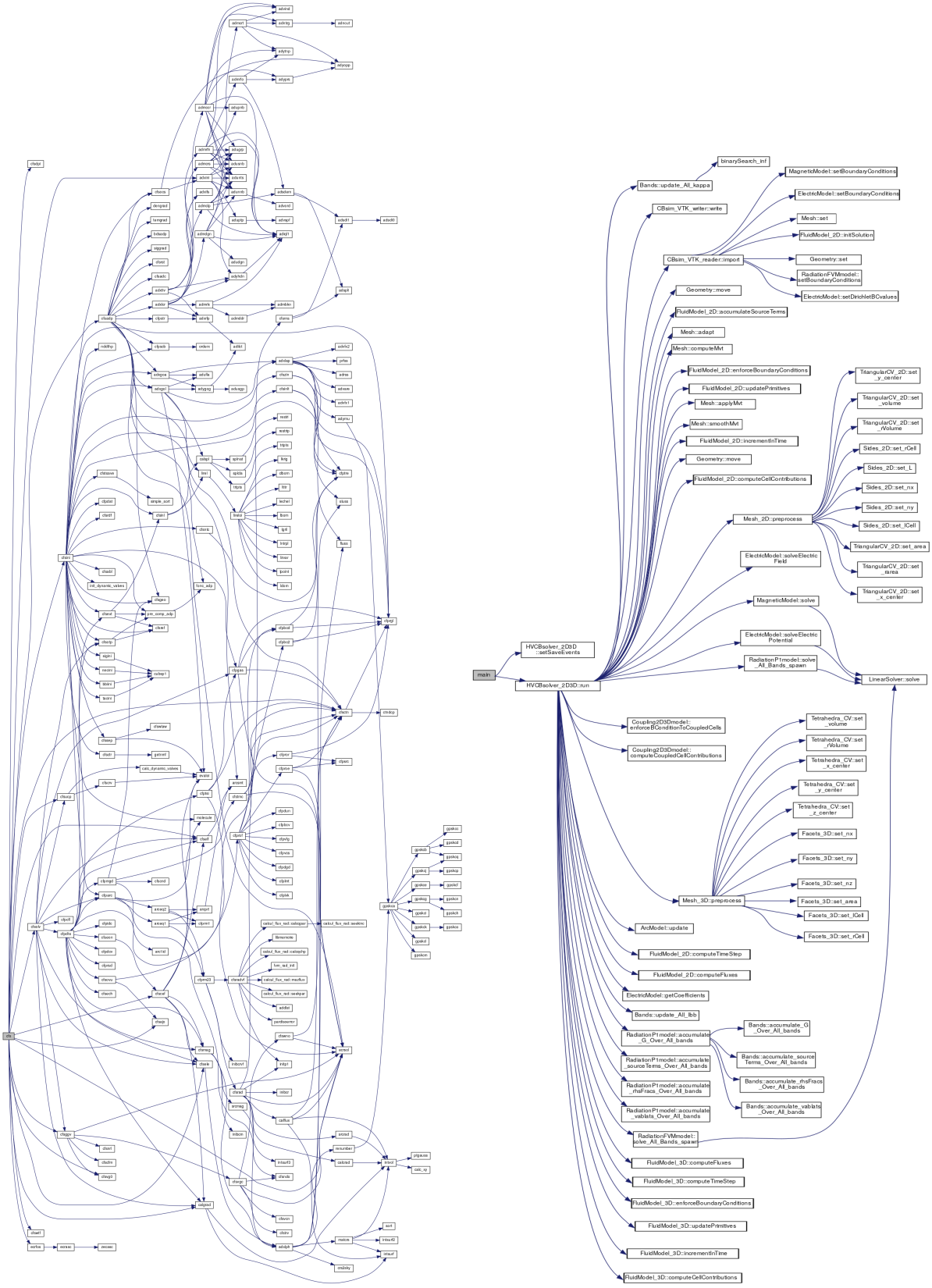


Figure 5.5 Aspect structurel : MC^3 original (à gauche) et la nouvelle conception (à droite)

5.5 Estimation des gains de performance

5.5.1 Protocole de test

Afin de comparer les résultats de performance en se plaçant dans une configuration proche de l'utilisation industrielle, une géométrie semblable à celle d'un disjoncteur haute-tension est utilisée (Figure 5.6). Une simulation de coupure d'un arc électrique de court-circuit de 15.7 millisecondes est réalisée avec un appareil à différentes positions d'ouverture. Trois positionnements sont choisis : après séparation ($+0.2\ ms$) ; à mi-course d'ouverture ($+6\ ms$) et à pleine course d'ouverture ($+12\ ms$). De cette façon, l'arc est de différentes tailles, les calculs des termes sources associés représentent donc des proportions variables de temps machine. Le domaine de calcul actif des termes sources croît avec les pas de temps. Pour les trois calculs, cinq microsecondes de simulation sont réalisées, soit environ 300 itérations pour mesurer le temps d'exécution et les maillages comportent environ cent-quinze-mille éléments. Avec ce court instant de simulation, l'adaptation de maillage ne transforme quasiment pas le maillage qui est presque immobile, les comparaisons entre la version de référence de MC^3 et la version optimisée sont donc représentatives. Les trois solutions initiales sont identiques entre la version de référence de MC^3 et la version optimisée.

5.5.2 Résultats

Calcul

Les trois simulations produisent des résultats similaires (Figure 5.6) entre la version de référence de MC^3 et la version optimisée. Cependant, une validation complète de l'outil sera requise et elle sera effectuée chez l'industriel lors de la phase d'industrialisation. En effet, de nouveaux cas industriels et tests unitaires seront nécessaires pour valider le nouvel outil dans son ensemble avec une configuration multi-physiques complète.

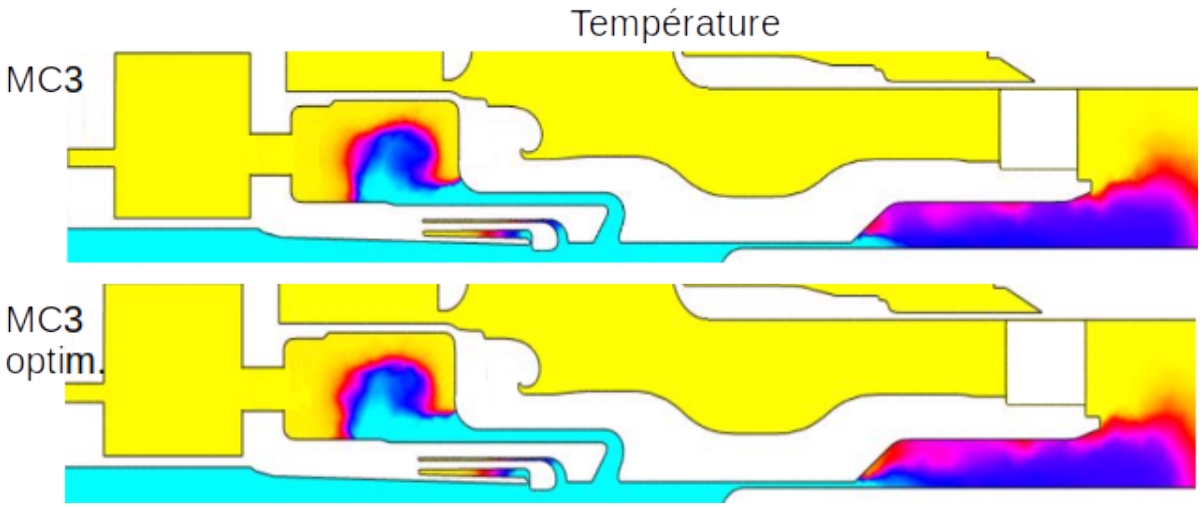


Figure 5.6 Résultats de la simulation multiphysiques à mi-ouverture (+6 ms)

Mémoire

Le tableau 5.1 présente des résultats généraux de mesures réalisées sur le même système que celui utilisé pour la phase d'analyse (Intel Core i7-6700k (4 cœurs, 4x 256Ko L2-8MB L3), 32 GB DDR4@2400 Mhz 15-15-15-35). La mémoire a été optimisée tant du point de vue de la taille utilisée (la plus grande réduction provient du retrait du solveur LU) que de l'efficacité des accès (tableau 5.1 et Figure 5.7).

Tableau 5.1 Récapitulatif d'éléments de performance

thèmes	Passé	Présent	Évolution
Usage DRAM	$\approx 900Mo$	$\approx 87Mo$	↘ 10×
Débit moyen DRAM	1.4 GB/s	10.1 GB/s	↗ 7×
Latence moyenne accès	27 cycles CPU	7 cycles CPU	↘ 4×

Remarque : L'usage mémoire est fortement réduit, avec les générations de CPU suivantes qui possèdent plus de cache (60 MB L3/CPU pour certains modèles !), un système NUMA@4 sockets pourrait être suffisant pour rendre le programme dit *"cache only"*, c'est à dire qu'aucun accès à la mémoire DRAM serait alors réalisé. Cela présente le potentiel d'une accélération supplémentaire très importante car toutes les données seraient conservées en cache et simultanément. Alors, les limitations dues à la latence de la DRAM seraient éliminées.

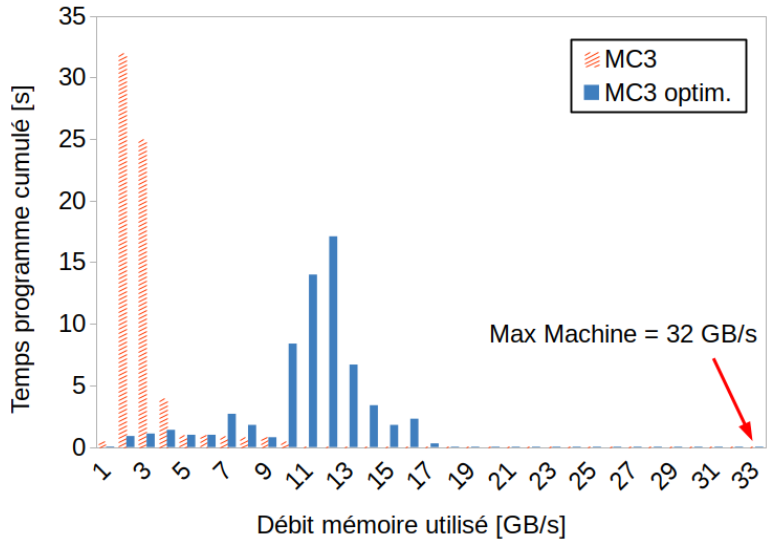


Figure 5.7 Temps passé en fonction de la bande passante mémoire

Temps CPU

D'après les vitesses d'exécution mesurées pour les trois instants de simulation calculés, le facteur global d'accélération entre la version de référence de MC^3 et la version optimisée, est estimé entre 30 à 100 fois, et dépend de la configuration géométrique et électrique du calcul. Les prochaines figures (5.8 et 5.10) montrent une nette amélioration vis-à-vis des mesures réalisées lors de l'analyse générale de MC^3 (section 4.3.1). La complexité algorithmique (Figure 5.8) est maintenant plus proche de $O(n\ log\ n)$.

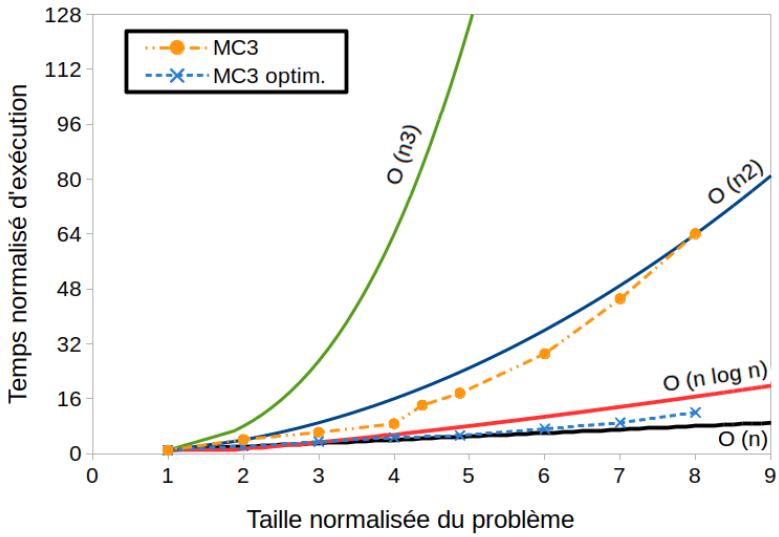


Figure 5.8 Optimisation : complexité algorithmique (maillage fixe)

Pour la suite des mesures, le système utilisé est Cedar (Compute Canada), dont la configuration est du type NUMA@4 sockets Intel Xeon E7-4809v4 (8 cores/socket, 8x 256Ko L2-20MB L3, DDR4@1866Mhz). Les temps de calcul sont relativement bien équilibrés entre chaque physique (Figure 5.9), ce qui signifie que, pour ce cas test, l'assignation du nombre de threads par physique minimise le temps de calcul global. Néanmoins, l'équilibrage de charge et l'affinité des threads devront être optimisés lors de la phase d'industrialisation afin d'être adaptés à un large spectre de configurations de calcul.

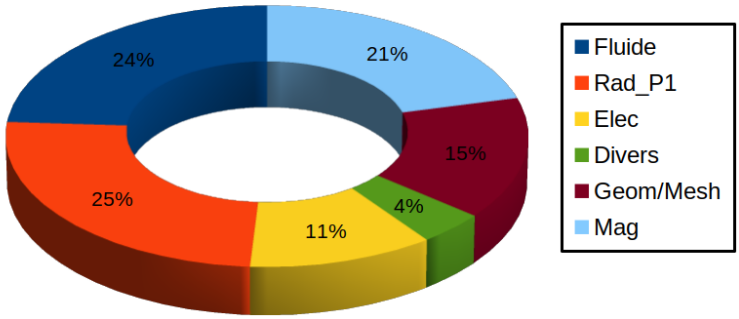


Figure 5.9 Répartition du temps de calcul en fonction des physiques

La scalabilité de la version parallèle de référence de MC^3 n'est pas très bonne : au-delà de six cœurs utilisés, un ralentissement de la vitesse d'exécution est observé. La figure 5.10, présente les améliorations apportées par la version optimisée de MC^3, dont la limite de scalabilité dépasse vingt cœurs. La limite de la scalabilité augmente avec la taille du problème.

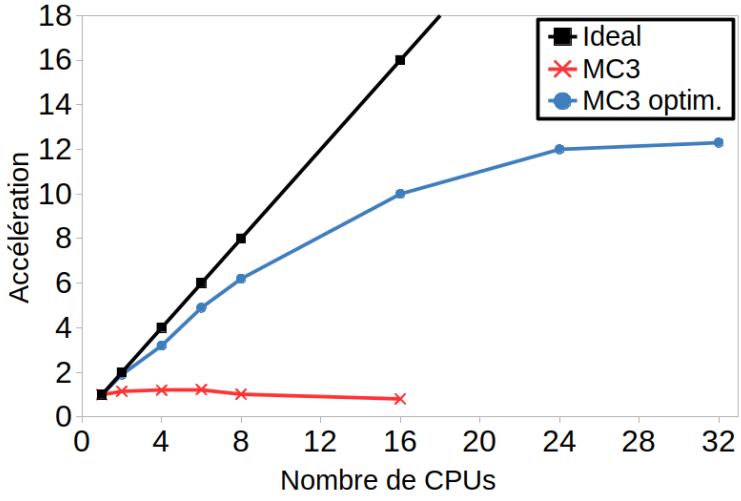


Figure 5.10 Scalabilité après optimisation (≈ 115 000 éléments)

CHAPITRE 6 INTÉGRATION D'UN NOUVEAU MODÈLE DE COUPLAGE DE MÉCANIQUE DES FLUIDES 2D AXISYMÉTRIQUE 3D

6.1 Motivation

La résolution en 3D a le potentiel d'améliorer la fidélité des résultats numériques versus les mesures d'essais. Cela renforce également la prise en compte des phénomènes à différentes échelles de temps et d'espace présents lors de l'opération d'un disjoncteur haute-tension.

Bien que la mise en place de calculs tri-dimensionnels soit de manière générale de plus en plus praticable et pratiquée, le temps nécessaire aux opérations de génération de maillage et de résolution, ainsi que l'impact financier associé aux matériels requis pour exécuter de larges simulations 3D restent coûteux. D'un point de vue industriel, les calculs sur des domaines pleinement 3D ne sont pas toujours justifiés. En effet, lorsqu'une grande partie du domaine est 2D axisymétrique par nature, ainsi que l'écoulement fluide associé, alors un calcul 3D complet n'est pas justifié. Cette remarque s'applique parfaitement au cas des simulations numériques des disjoncteurs haute-tension où l'utilisation de parties 3D en complément des parties 2D axisymétriques classiques représenterait un excellent compromis en pratique. En effet, la précision des simulations peut être améliorée tout en limitant le coût global de la simulation complète. Pour réaliser ce couplage, et améliorer la fidélité des solutions de MC^3, nous avons donc besoin d'une méthode pour coupler des calculs 3D avec des calculs 2D axisymétriques au sein d'un seul et même outil de simulation. Dans ce chapitre, on propose une nouvelle méthode de couplage fluide capable d'être intégrée dans un unique solveur arbitrairement 2D-3D, et ainsi de répondre aux besoins des simulations numériques de l'opération des disjoncteurs haute-tension.

6.2 Les méthodologies de couplage existantes

Plusieurs auteurs ont proposé des couplages multi-dimensionnels tels que Zou *et al.* [134] qui simulent les turbines sous enveloppe, en étendant des modèles 1D vers des distributions 2D à une interface de couplage par une méthode de mise à l'échelle. Chen *et al.* [21] proposent de coupler des domaines 2D et 3D pour la simulation des écoulements de rivière avec une méthode FSC (**F**ree **S**urface **C**orrection) basée sur une procédure prédicteur-correcteur. Cependant, cette approche est difficile à adapter à d'autres modélisations physiques. Dans la littérature, d'autres méthodes existent pour des couplages en général pour le même nombre de dimensions. Parmi les méthodes très pratiquées de couplages de type spatial ou multi-

fidélités (voir classification section 2.3) qui pourraient nous inspirer, on retrouve les techniques de maillages superposés chimères. Des interpolations sont alors pratiquées entre les différents maillages pour obtenir une solution locale unique. Ceci se prête bien par exemple aux solutions 2D-2D ou 3D-3D, cependant dans le cas 2D-3D, une simple interpolation de la solution entre les maillages ne peut être réalisée : quelle est la solution prédominante/la plus réaliste, la solution 2D ou la solution 3D ? Les techniques chimères sont également largement utilisées pour les problèmes multiphysiques couplés avec la mécanique des fluides, par exemple : Ryan et al. [110] couplent la CFD avec les calculs acoustiques (CAA), Miller et al. [77] réalisent un couplage Fluide-Structure (FSI), Schwartz et al. [113] couplent les calculs d'ailes d'avion avec celui des nacelles supportant les moteurs, Wang et al. [126] gèrent des corps mobiles. Les simulations des composants de turbomachines considèrent généralement un seul passage ou secteur de distributeurs, des interfaces d'interpolation sont alors nécessaires. Ces interpolations sont réalisées avec des techniques à base de General Grid Interface (GGI), telle que l'interface *mixingPlane* notamment intégrée à Fluent [6] et à OpenFOAM puis améliorée par Beaudoin et al. [9]. L'inconvénient majeur de ces méthodes est qu'elles ne sont pas conservatives, des moyennages sont effectués au niveau des interfaces, lissant également la solution en ajoutant de la diffusion numérique. Dans MC^3, on a besoin de la capacité de transférer des chocs à travers les interfaces de couplage (dont les directions ne sont pas connues *a priori*). A notre connaissance, aucune méthode de couplage de calcul d'écoulement fluide compressible entre des régions 3D et 2D axisymétriques proposée dans la littérature ne peut répondre aux besoins de couplage pour la simulation numérique de l'opération des disjoncteurs haute-tension dans MC^3. Nous devons donc développer une nouvelle méthode de couplage dont les pré-requis sont énoncés à la section suivante.

6.3 Considérations générales et description des prérequis du modèle de couplage

6.3.1 Hypothèses et pré-requis

D'après les caractéristiques de l'écoulement fluide, le couplage doit :

- autoriser le transfert d'information de manière bi-directionnelle, c.à.d. pour des directions d'écoulement de la partie 2D axisymétrique vers la partie 3D, et inversement, puisque la direction de l'écoulement n'est pas connue *a priori* et peut changer en fonction du temps et/ou de la position.
- fonctionner pour des écoulements subsonique ou supersonique à travers l'interface, car l'écoulement peut être hautement compressible et supersonique dans certaines régions.
- respecter la conservation de la masse, de la quantité de mouvement et de l'énergie.

— être réalisé entre des espaces de coordonnées 2D axisymétriques $(\vec{r}, \vec{\theta}, \vec{z})$ et 3D $(\vec{x}, \vec{y}, \vec{z})$

6.3.2 Choix d'implémentation réalisés

L'hypothèse d'un fluide non visqueux est supposée :

— Équations d'Euler.

La méthode des volumes finis (FVM) est utilisée :

— La méthode de couplage doit être compatible avec la méthode des volumes finis.

6.4 Méthodologie de couplage 2D axisymétrique 3D proposée

6.4.1 Philosophie du couplage

Dans le but de coupler des régions 3D et 2D axisymétriques, on propose de coupler les flux numériques 3D et 2D échangés entre les volumes de contrôle adjacents reposant sur une interface de couplage. Dans la méthode des volumes finis, sans perte de généralité, les cellules sont toujours traitées en tant que volumes de contrôle tri-dimensionnels, ainsi nous pouvons considérer les cellules 2D axisymétriques dans la zone de couplage comme des cellules 3D, comme exposé à la figure 6.1. Ces volumes sont obtenus par révolution de la surface 2D des cellules autour de l'axe de symétrie $(r = 0)$.

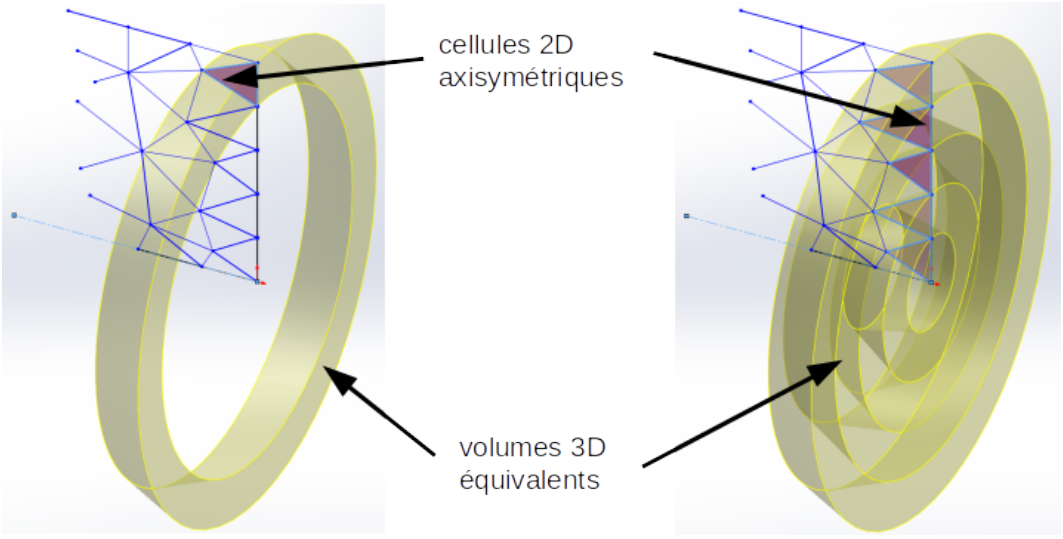


Figure 6.1 Traitement des cellules 2D axisymétriques couplées en tant que volumes 3D

La révolution de l'ensemble des côtés couplés des cellules 2D autour de l'axe de symétrie génère une surface, pas nécessairement plane, définissant l'interface de couplage. De part et d'autre de cette interface, reposent les cellules utilisées pour calculer les flux numériques. Les

faces couplées des volumes 3D, de formes arbitraires, s'appuient sur l'interface de couplage, telle qu'illustrée à la figure 6.2.

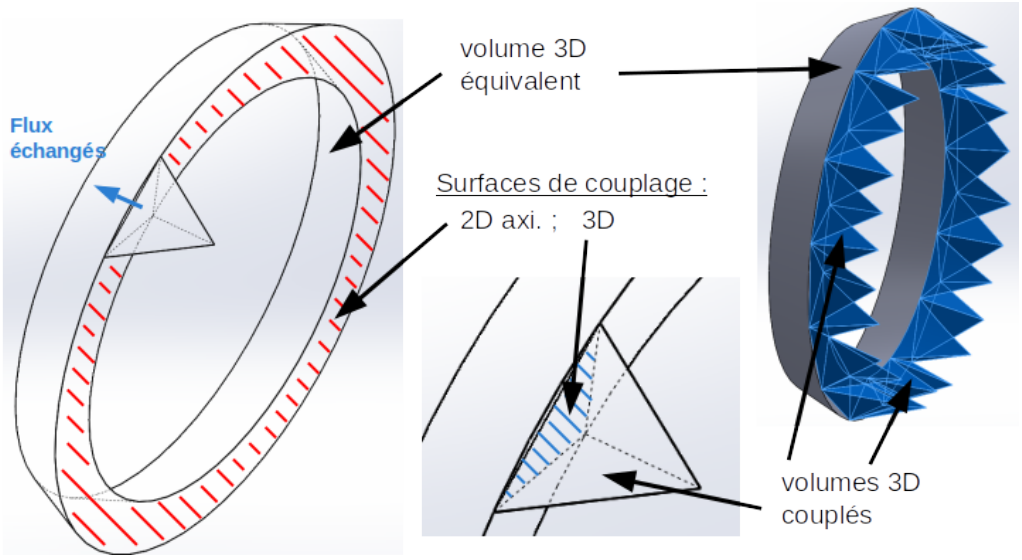


Figure 6.2 Couplage tétraèdres (3D) <-> triangles (2D axisymétrique)

Les flux (un pour chaque équation dans le système (3.1)/(6.1)) sont échangés entre les cellules voisines à travers leur surface de contact. Ainsi, de multiples cellules 3D voisines pourront contribuer à une seule cellule 2D axisymétrique (voir Figure 6.2), selon des critères de conformité de maillage qui seront détaillés à la section 6.4.3.

Comme on peut le remarquer, les cellules 3D doivent échanger cinq flux à travers les faces de leur volume, alors que les cellules 2D axisymétriques en échangent seulement quatre car il y a une équation de moins pour la conservation de la quantité de mouvement. Dans la prochaine section, on propose une méthode de prise en compte de cette différence dans le nombre de flux.

6.4.2 Système d'équations

Le système d'équations d'Euler décrivant un écoulement de fluide compressible 2D axisymétrique et non visqueux est le système (3.1) précédemment introduit. L'extension de ce système d'équations au cas tri-dimensionnel dans un espace cartésien (x,y,z) donne :

$$\frac{\partial \vec{U}}{\partial t} + \frac{\partial \vec{F}_x}{\partial x} + \frac{\partial \vec{F}_y}{\partial y} + \frac{\partial \vec{F}_z}{\partial z} = \vec{0} \tag{6.1}$$

où,

$$\vec{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix}, \quad \vec{F_x} = \begin{bmatrix} \rho u \\ \rho u^2 + P \\ \rho uv \\ \rho uw \\ u(\rho E + P) \end{bmatrix}, \quad \vec{F_y} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + P \\ \rho vw \\ v(\rho E + P) \end{bmatrix}, \quad \vec{F_z} = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + P \\ w(\rho E + P) \end{bmatrix}$$

avec ρ la masse volumique, P la pression, $\vec{W} = u.\vec{x} + v.\vec{y} + w.\vec{z}$ le vecteur vitesse et $E = e + \frac{1}{2}||W||^2$ l'énergie totale où e est l'énergie interne.

Le système d'équations 2D (3.1) et le système d'équations 3D (6.1) sont résolus indépendamment par la méthode des volumes finis (FVM) basée sur le schéma numérique de Roe [106] précédemment décrite dans la section 3.2.1. Les lois de conservation sont intégrées sur chaque cellule/volume pour obtenir l'ensemble discret d'équations. Le champ des variables est interprété comme une valeur moyenne sur chaque volume. Lors de la procédure de calcul, des flux numériques sont échangés à travers les faces des volumes finis adjacents, qui peuvent être de n'importe quel type.

Cependant, il y a toujours une disparité dans le nombre de flux à traiter, puisque le système (6.1) possède une équation de plus que l'équivalent 2D axisymétrique pour la conservation de la quantité de mouvement. De plus, l'équation de conservation d'énergie est également différente car l'expression du terme d'énergie cinétique est différente. La solution la plus aisée serait d'ignorer une des équations de quantité de mouvement lors de l'échange des flux, cependant cette solution n'est pas totalement conservatrice.

Pour des écoulements simples, cette portion peut être ignorée mais pour des écoulements plus complexes, cette portion n'est plus négligeable, et la totalité de la quantité de mouvement doit être transférée d'une certaine manière. Ce qui conduit à échanger totalement les cinq flux 3D avec les cellules 2D axisymétriques.

Nous proposons alors de considérer la troisième composante de vitesse dans le système d'équations 2D axisymétriques (spatialement appliquée uniformément dans tout le domaine 2D dans notre cas), le but étant d'obtenir le même nombre d'équations que le système 3D d'une part, et de supporter des écoulements complexes d'autre part. Cette nouvelle composante de vitesse sera transportée dans le domaine de la même manière que la convection d'une quantité scalaire.

Afin d'étendre le système d'équations 2D axisymétriques, on s'inspire de la procédure de Champmartin *et al.* [11], qui traitent des écoulements multi-matériaux avec symétrie cy-

lindrique. On exprime tout d'abord les équations d'Euler 3D en coordonnées cylindriques (r, θ, z) et on pose la symétrie cylindrique ($\frac{\partial ...}{\partial \theta} = 0$), le but étant de découpler une des trois équations de quantité de mouvement des autres équations du système (6.1). Ce qui conduit au système d'équations :

$$\frac{\partial \rho}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}(r\rho u_r) + \frac{\partial}{\partial z}(\rho u_z) = 0 \tag{6.2}$$

$$\frac{\partial(\rho u_z)}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}(r\rho u_z u_r) + \frac{\partial}{\partial z}(\rho u_z^2 + P) = 0 \tag{6.3}$$

$$\frac{\partial(\rho u_r)}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}(r(\rho u_r^2 + P)) + \frac{\partial}{\partial z}(\rho u_z u_r) = \frac{1}{r}(\rho u_\theta^2 + P) \tag{6.4}$$

$$\frac{\partial(\rho u_\theta)}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}(r\rho u_\theta u_r) + \frac{\partial(\rho u_\theta u_z)}{\partial z} = -\frac{1}{r}\rho u_\theta u_r \tag{6.5}$$

$$\frac{\partial(\rho E)}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}(r\rho u_r(E + \frac{P}{\rho})) + \frac{\partial}{\partial z}(\rho u_z(E + \frac{P}{\rho})) = 0 \tag{6.6}$$

Avec E l'énergie totale, composée de l'énergie interne e et de l'énergie cinétique $\frac{1}{2}\|u\|^2$. Le système est fermé en utilisant une équation d'état.

Les équations dans ce nouvel ensemble sont couplées par l'intermédiaire de l'équation d'énergie (6.6), où $E = e + \frac{1}{2}\|u_r\|^2 + \frac{1}{2}\|u_\theta\|^2 + \frac{1}{2}\|u_z\|^2$, il faut alors les rendre indépendantes. On remplace le terme d'énergie totale E dans l'équation d'énergie par l'énergie totale partielle E^* de la forme :

$$E^* = e + \frac{1}{2}\|u_r\|^2 + \frac{1}{2}\|u_z\|^2 = E - \frac{1}{2}\|u_\theta\|^2 \tag{6.7}$$

Lorsqu'on remplace E par E^* dans l'équation (6.6), on a :

$$\begin{aligned} &\frac{\partial(\rho E^*)}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}(r\rho u_r(E^* + \frac{P}{\rho})) + \frac{\partial}{\partial z}(\rho u_z(E^* + \frac{P}{\rho})) = \\ &\frac{\partial(\rho(E - \frac{u_\theta^2}{2}))}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}(r\rho u_r(E - \frac{u_\theta^2}{2} + \frac{P}{\rho})) + \frac{\partial}{\partial z}(\rho u_z(E - \frac{u_\theta^2}{2} + \frac{P}{\rho})), \end{aligned} \tag{6.8}$$

soit :

$$\begin{aligned} &\frac{\partial(\rho E^*)}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}(r\rho u_r(E^* + \frac{P}{\rho})) + \frac{\partial}{\partial z}(\rho u_z(E^* + \frac{P}{\rho})) = \\ &\frac{\partial(\rho E)}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}(r\rho u_r(E + \frac{P}{\rho})) + \frac{\partial}{\partial z}(\rho u_z(E + \frac{P}{\rho})) \\ &-\frac{\partial(\rho(\frac{u_\theta^2}{2}))}{\partial t} - \frac{1}{r}\frac{\partial}{\partial r}(r\rho u_r(\frac{u_\theta^2}{2})) - \frac{\partial}{\partial z}(\rho u_z(\frac{u_\theta^2}{2})) \end{aligned} \tag{6.9}$$

D'après l'équation (6.6), la deuxième ligne de l'équation (6.9) est nulle, cela se simplifie donc en :

$$\begin{aligned}\frac{\partial(\rho E^*)}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}(r\rho u_r(E^* + \frac{P}{\rho})) + \frac{\partial}{\partial z}(\rho u_z(E^* + \frac{P}{\rho})) = \\ -\frac{\partial(\rho(\frac{u_\theta^2}{2}))}{\partial t} - \frac{1}{r}\frac{\partial}{\partial r}(r\rho u_r(\frac{u_\theta^2}{2})) - \frac{\partial}{\partial z}(\rho u_z(\frac{u_\theta^2}{2}))\end{aligned} \tag{6.10}$$

En utilisant l'équation (6.5) et en y remplaçant u_θ par $\frac{u_\theta^2}{2}$ on obtient :

$$-\frac{\partial(\rho(\frac{u_\theta^2}{2}))}{\partial t} - \frac{1}{r}\frac{\partial}{\partial r}(r\rho u_r(\frac{u_\theta^2}{2})) - \frac{\partial}{\partial z}(\rho u_z(\frac{u_\theta^2}{2})) = \frac{\rho u_r u_\theta^2}{r} \tag{6.11}$$

Nous obtenons alors la nouvelle forme de l'équation d'énergie :

$$\frac{\partial(\rho E^*)}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}(r\rho u_r(E^* + \frac{P}{\rho})) + \frac{\partial}{\partial z}(\rho u_z(E^* + \frac{P}{\rho})) = \frac{\rho u_r u_\theta^2}{r} \tag{6.12}$$

Ainsi, on peut maintenant séparer le système d'équations en deux sous-systèmes. Le premier sous-système inclut les équations : (6.2), (6.4), (6.3) et (6.12), tandis que le second système est constitué seulement de l'équation (6.5), qui est l'équation de transport pour u_θ. Le premier sous-système est indépendant de u_θ, excepté pour les équations (6.4), (6.5) et (6.12) où de nouveaux termes sources apparaissent.

Finalement, le premier sous-système s'écrit :

$$\frac{\partial \vec{U^*}}{\partial t} + \frac{1}{r}\frac{\partial \vec{F_r^*}}{\partial r} + \frac{\partial \vec{F_z^*}}{\partial z} = \vec{S^*} \tag{6.13}$$

où,

$$\vec{U^*} = \begin{bmatrix}\rho \\ \rho u_z \\ \rho u_r \\ \rho E^*\end{bmatrix}, \vec{F_r^*} = \begin{bmatrix}r\rho u_r \\ r\rho u_z u_r \\ r(\rho u_r^2 + P) \\ r u_r(\rho E^* + P)\end{bmatrix}, \vec{F_z^*} = \begin{bmatrix}\rho u_z \\ \rho u_z^2 + P \\ \rho u_z u_r \\ u_z(\rho E^* + P)\end{bmatrix}, \vec{S^*} = \begin{bmatrix}0 \\ 0 \\ P/r \\ 0\end{bmatrix} + \begin{bmatrix}0 \\ 0 \\ (\rho u_\theta^2)/r \\ (\rho u_\theta^2 u_r)/r\end{bmatrix} + \vec{S_{arc}}$$

avec ρ la masse volumique, P la pression, $\vec{V} = (u_r, u_z)^T$ le vecteur vitesse et $E^* = e + \frac{1}{2}||V||^2$ l'énergie totale où e est l'énergie interne.

En comparaison avec le système d'équations 2D axisymétrique (3.1), le premier sous-système (6.13) à résoudre diffère seulement par la présence de termes source supplémentaires. Ainsi, ce sont les mêmes flux que ceux du cas pleinement 2D axisymétrique qui doivent être calculés, avec le seul ajout de ces termes sources (dus uniquement à la présence de u_θ). Quand u_θ

s'annule, $E^* = E$ et les équations résolues deviennent identiques à 3.1.

Concernant le second sous-système (équation 6.5), la quantité ρu_θ est transportée dans le domaine à la vitesse de convection $\vec{V}(u_r, u_z)$. D'un point de vue numérique, une procédure amont est utilisée pour calculer ces flux convectifs.

Finalement, l'intégration des équations sur les volumes de contrôle Ω conduit au schéma volumes finis :

$$U^{*,n+1} = U^{*,n} - \frac{\Delta t}{\Delta \Omega} \int_{\partial\Omega} (\vec{F_r^*},\ \vec{F_z^*})^T .\ \vec{n}\ dS_\Omega - \Delta t \vec{Q}^{*,n}, \tag{6.14}$$

et

$$U_\theta^{n+1} = U_\theta^n - \frac{\Delta t}{\Delta \Omega} \int_{\partial\Omega} (\vec{F_{\theta,r}},\ \vec{F_{\theta,z}})^T .\ \vec{n}\ dS_\Omega - \Delta t \vec{Q}_\theta^n, \tag{6.15}$$

Avec $\vec{n} \in \mathbb{R}^2$, la normale unitaire sortante et S_Ω les surfaces des volumes de contrôle. Les flux F_θ sont obtenus par une procédure amont dont la direction de propagation dépend de la solution de Roe (méthode découplée de Larrouturou [60]). De plus, les termes source Q sont intégrés sur les volumes de contrôle en utilisant une discrétisation centrée sur les cellules et sont considérés constants par élément.

Dans la procédure de résolution, le calcul des flux reste indépendant, comme pour une formulation volumes finis classique. Seule une condition limite spécifique est introduite pour signaler la position de l'interface de couplage entre les domaines de différentes dimensions. Puis, les flux couplés entre les cellules 2D axisymétriques et les cellules 3D sont calculés en utilisant les états de part et d'autre, écrits dans le même repère de coordonnées (r, θ, z), donc, ayant le même nombre de composantes de vitesse. De la sorte, la condition de conservation de la méthode de couplage est renforcée. Les cellules 3D couplées vont contribuer aux cellules 2D axisymétriques adjacentes dans des proportions partielles relatives à leurs surfaces de contact (d'échange) si le maillage n'est pas conforme, ou, complètes si le maillage est conforme. Ces considérations de maillage sont décrites dans la section suivante. Le nouveau système d'équation 2D axisymétriques a aussi l'avantage de pouvoir traiter les cas des arcs tournants dans les simulations MC^3, ce qui était impossible avec la formulation axisymétrique originale (3.1). De plus, la nouvelle méthode est avantageusement indépendante du schéma numérique de calcul des flux (Roe, AUSM, HLL...).

6.4.3 Aspects géométriques et traitement des flux

Conformité de maillage

Par conception des algorithmes de la méthode de couplage proposée, aucune superposition de maillage n'est nécessaire. C'est un avantage vis-à-vis des méthodes de type chimère qui

peuvent s'avérer moins précises et moins rapides. En effet, dans les régions où les maillages se superposent, plus de calculs sont réalisés car les domaines de calcul sont plus étendus et la solution est ensuite localement interpolée entre les deux maillages. La conservation de toutes les propriétés fluides n'est alors possible qu'avec des interpolations conservatives.

La figure 6.3 illustre la définition au sens du couplage de la conformité de maillage. Une cellule 3D peut contribuer à une seule cellule 2D dans le cas où le maillage est conforme ou à plusieurs s'il est non-conforme. Dans ce dernier cas, le coefficient $0 < \beta_{i\to k} \leq 1$ permet alors de prendre en compte le ratio de la surface $S_{i\ f3D}$ de la facette 3D, i, effectivement en échange avec la cellule 2D, k. Son expression est :

$$\beta_{i\to k} = \frac{S_{k\ f2D} \bigcap S_{i\ f3D}}{S_{i\ f3D}}.$$

Lorsque le maillage est conforme $\beta_{i\to k} = 1$. Étant donné la facétisation 3D, la somme des surfaces des facettes 3D n'est pas égale à la surface 2D axisymétrique. Dans le cas d'un écoulement dans un canal, la section de passage du fluide varie brutalement. Pour assurer la conservation du débit, le facteur α_k de mise à l'échelle est appliqué aux flux surfaciques tel que :

$$\alpha_k = \frac{S_{k\ f2D}}{\sum_i^{n_f 3D} (\beta_{i\to k} \times S_{i\ f3D})}$$

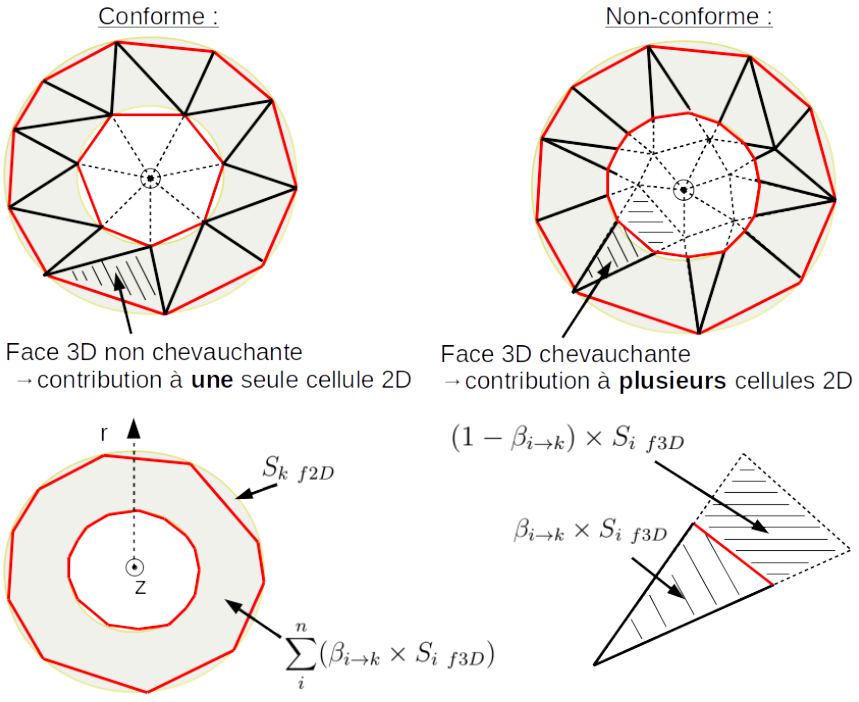


Figure 6.3 Conformité des maillages au sens du couplage

Finalement, la contribution d'un élément 3D vers un élément 2D axisymétrique s'écrit :

$$contribution = \alpha_k \times \sum_{i}^{n_f 3D} (\beta_{i \to k} \times S_{i\ f3D} \times F_{i \to k})$$

Pour l'élément 3D, cette contribution est simplement retranchée (traitement FVM standard). Pour les couplages axiaux le même principe de calcul est appliqué.

Cette souplesse dans la gestion des maillages permet d'utiliser les maillages de MC^3 sans contrainte spécifique. La création, la gestion des maillages 2D et 3D et la résolution des équations 2D et 3D (mise à part dans la région couplée) sont toutes des opérations indépendantes. La correspondance des éléments voisins 2D-3D est à créer et maintenir à jour si les maillages sont adaptatifs et/ou mobiles. L'efficacité globale et la souplesse d'utilisation de la méthode de couplage sont donc excellentes. Il est facilement possible d'étendre la méthode de couplage à d'autres physiques modélisées en utilisant la méthode des volumes finis.

Maillages utilisés pour la validation

La figure 6.4, illustre une partie des maillages que nous avons utilisés pour valider la méthode de couplage. En pratique, les volumes de contrôle qui sont les plus proches de l'axe doivent être raffinés pour prendre en compte une capture de la géométrie et/ou du maillage 2D correcte telle que les maillages 2 et 4 sur la figure 6.4.

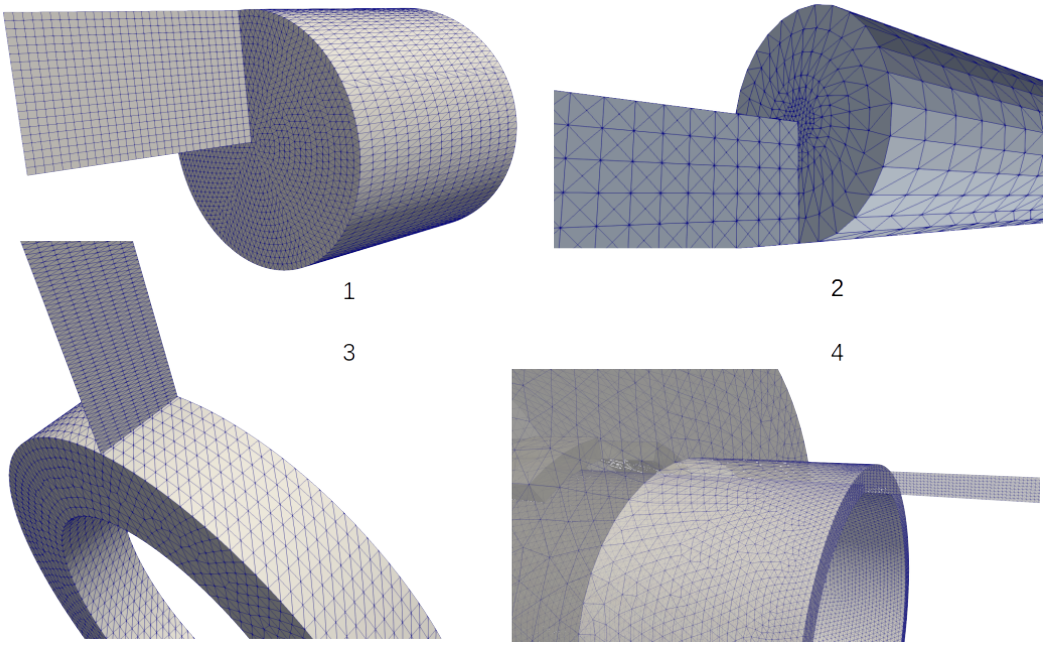


Figure 6.4 Maillages pour le couplage

Caractéristiques des maillages :

- le maillage n° 1 comporte 600(**a**)/9600(**b**) cellules cartésiennes et 121 000(**a**)/341 000(**b**) tétraèdres environ, la taille spécifique des côtés est de $h = 0.01\ m$ / $h = 0.0025\ m$,
- le maillage n° 2 est constitué de 2000 triangles et d'environ 125 000 tétraèdres pour une taille caractéristique de $h = 0.04\ m$,
- le maillage n° 3, utilisé pour les couplages radiaux, comporte 2000 triangles et 72 000 tétraèdres pour une taille de $h = 0.05\ m$,
- le maillage n° 4 est prévu pour le cas de vérification sur géométrie semi-industrielle avec 2000 triangles et 192 000 tétraèdres.

Il y sera fait référence dans la prochaine section où les résultats de validation sont présentés.

6.5 Résultats et discussion

Dans cette section, on présente uniquement les résultats pour la version de couplage utilisant des maillages conformes. Afin de faciliter les étapes de vérification de la méthode de couplage, dans tout ce chapitre, on utilise l'équation des gaz parfaits avec la constante du gaz $R = 287.0\ J.kg^{-1}.K^{-1}$.

6.5.1 Couplage axial

Écoulement à vitesse nulle - non-tourbillonnant ($u_\theta = 0.0$)

Le premier cas test est un écoulement stationnaire et la géométrie est un tube de diamètre $0.4\ m$, où la vitesse de l'écoulement est nulle (on n'impose aucune vitesse à l'entrée). Une région 2D axisymétrique est couplée avec une région 3D, toutes deux de longueur $L = 20\ m$. Le maillage n^o 2 est utilisé. Ce cas test permet de s'assurer qu'aucun écoulement n'est généré à l'interface de couplage.

Les conditions aux limites sont décrites sur la figure suivante :

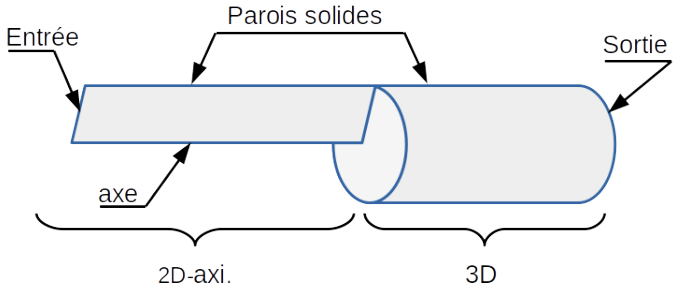


Figure 6.5 Conditions aux limites - cas à vitesse nulle

Conditions intérieures

$\rho\ [kg.m^{-3}]$	$V\ [m.s^{-1}]$	$T\ [K]$	$P\ [Pa]$
10.0	0.0	300.0	861,000

Plusieurs milliers d'itérations sont réalisées pour vérifier si le couplage reste conservateur (l'écoulement doit rester constant et ne pas créer/perdre de la masse ou de l'énergie). Les résultats de la figure 6.6 confirment la conservation, après 200 000 itérations, de petites variations de l'erreur ε autour de la valeur nulle (à la précision arithmétique de la machine près) apparaissent. Ceci confirme (à ε près) qu'aucune création de masse ni de vitesse n'intervient lors de l'utilisation du couplage.

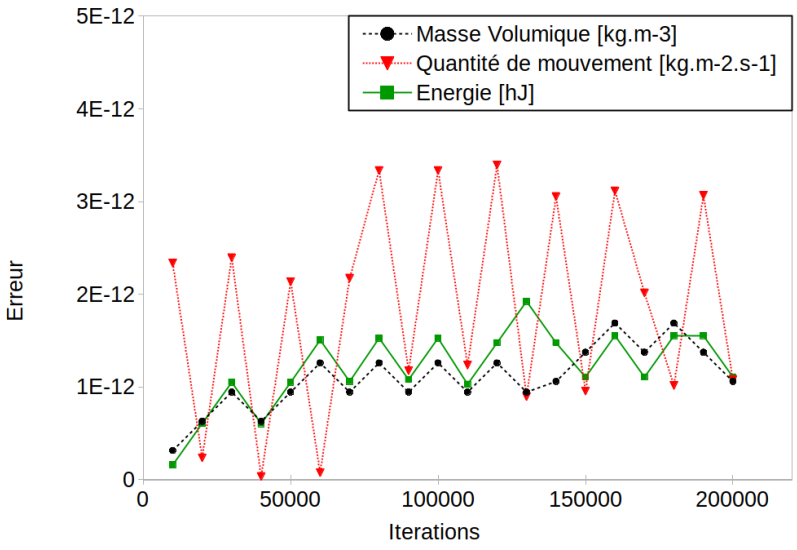


Figure 6.6 Conservation des quantités dans le domaine de calcul

Écoulement à vitesse constante (non nulle) - non-tourbillonnant ($u_\theta = 0.0$)

Ce cas stationnaire utilise la même géométrie et conditions aux limites que le cas précédent, mais on impose maintenant des conditions de pression à l'entrée et la sortie pour générer un écoulement :

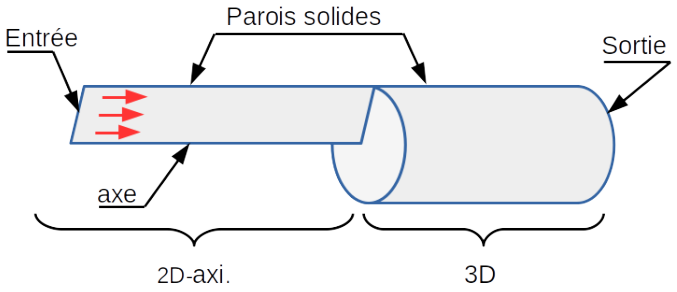


Figure 6.7 Conditions aux limites - cas à vitesse constante

Une vérification de la différence de débit massique et de débit d'énergie entre l'entrée et la sortie du domaine est réalisée. Le débit de masse/énergie entrant doit être égal (à une déviation ε près) au débit de masse/énergie sortant, l'écart est présenté à la figure 6.8.

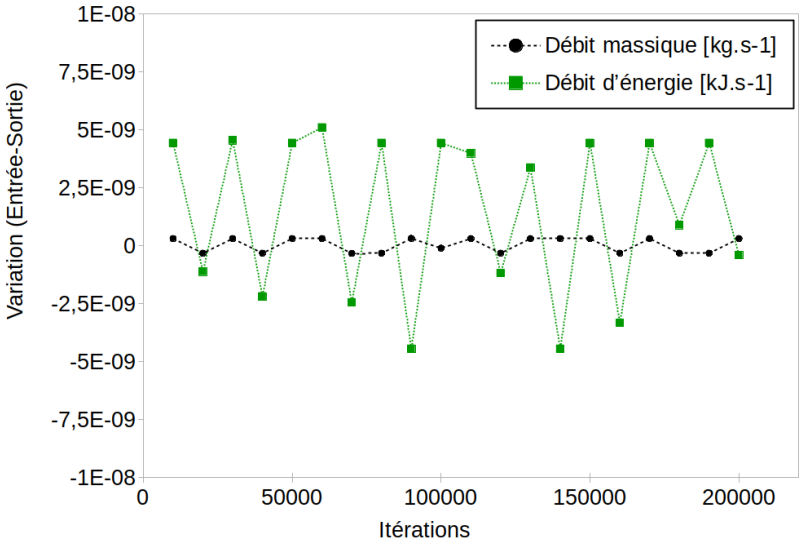


Figure 6.8 Écarts des débits massique et énergétique entre l'entrée et la sortie

Les écarts des débits massique et énergétique entre l'entrée et la sortie restent très petits au cours des 200 000 itérations, confirmant qu'aucune masse ou énergie n'est créée (à ε près) en utilisant le couplage avec une vitesse constante. Les très petites déviations sont attribuables aux erreurs de calcul en virgule flottante.

Transport d'un tourbillon à vitesse constante ($u_\theta = 10.0\ m.s^{-1}$)

Le transport d'un tourbillon constant par un champ de vitesse constante permet de vérifier que u_θ est transporté correctement entre les domaines 2D axisymétrique et 3D. De plus, la géométrie et la position finales du tourbillon sont prédictibles puisque le champ de transport est constant. La même géométrie et une configuration identique sont toujours utilisées, mais une région carrée (un tore par révolution autour de l'axe de symétrie) est initialisée à la vitesse azimutale constante $u_\theta = 10.0\ m.s^{-1}$. Le maillage n^o 1a est utilisé. La figure 6.9 et le tableau suivant fournissent les détails de la configuration.

Configuration de l'écoulement

$\rho\ [kg.m^{-3}]$	$V\ [m.s^{-1}]$	$T\ [K]$	$P\ [Pa]$	$u_\theta\ [m.s^{-1}]$
1.0	30.0	300.0	86, 100	10.0

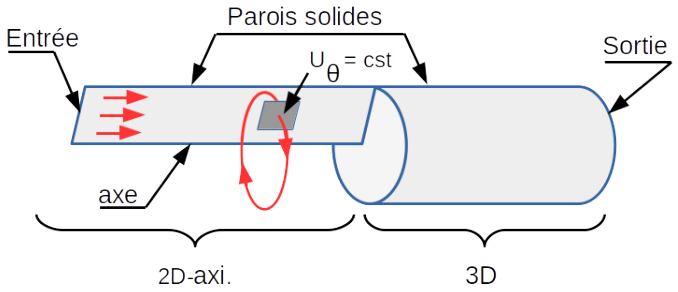


Figure 6.9 Configuration pour le transport u_θ du domaine 2D axisymétrique au domaine 3D

Après le transport, le *tore* ne devrait pas être déformé (à la diffusion numérique près) et devrait avoir conservé l'amplitude de u_θ constant.

La figure 6.10 indique que le profil de vitesse u_θ est bien conservé malgré le fait que la diffusion numérique le lisse quelque peu. Ainsi, la méthode de couplage fonctionne bien pour un écoulement de fluide en provenance de la région 2D axisymétrique.

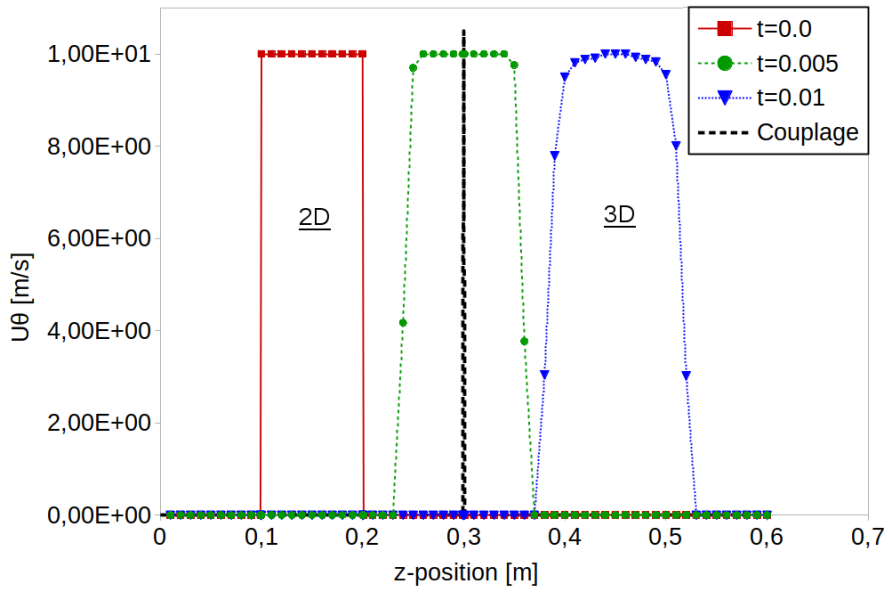


Figure 6.10 Transport de u_θ du domaine 2D axisymétrique au domaine 3D, vue du profil de vitesse u_θ le long de $\vec{z}$ à différents temps : $t = 0.0$, $t = 0.005$ et $t = 0.01$

La figure 6.11 montre les résultats de la simulation durant le transfert à travers le couplage, où l'on peut constater une transmission correcte de la vitesse u_θ à travers l'interface.

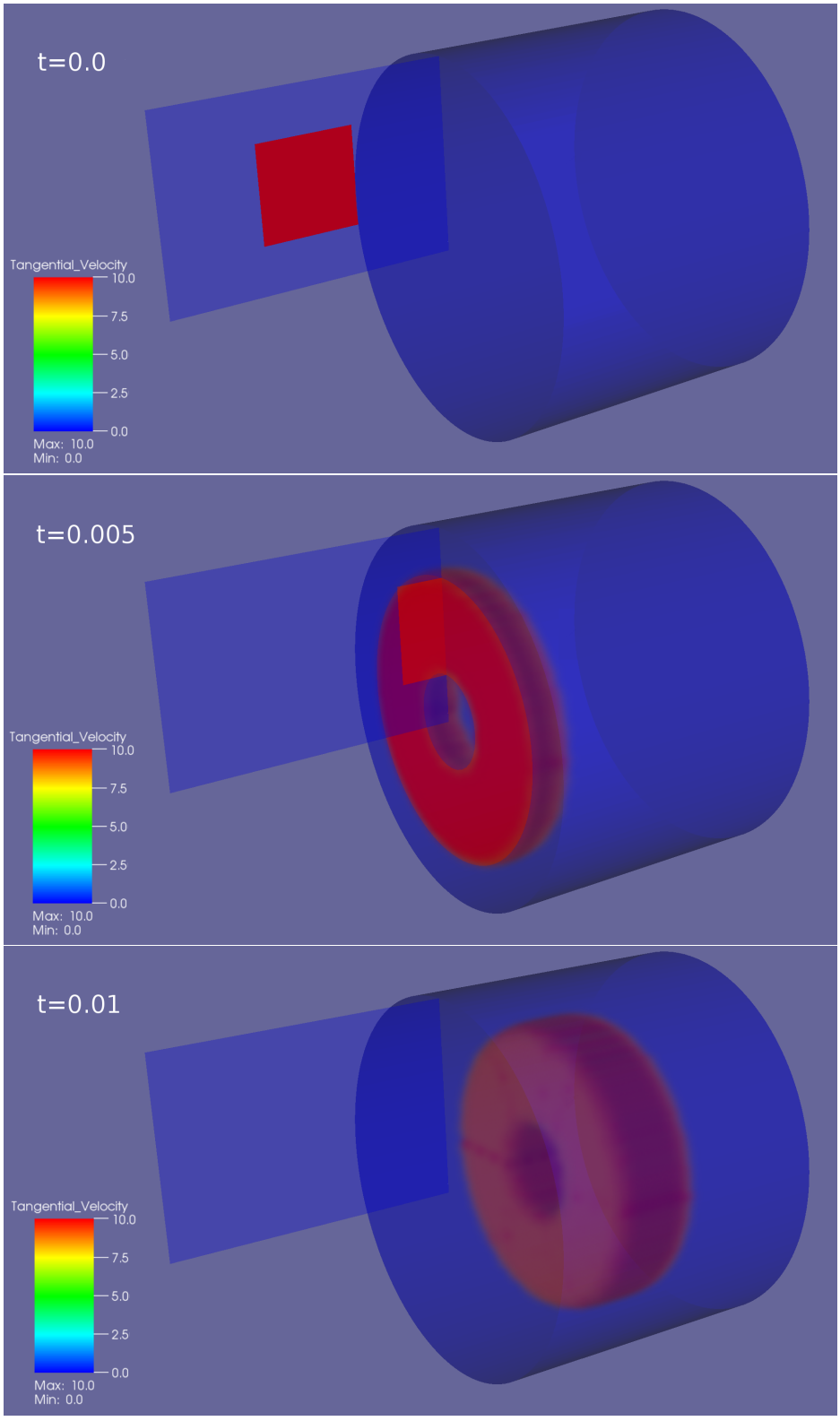


Figure 6.11 Transport de u_θ du domaine 2D axisymétrique au domaine 3D, vue du champ de vitesse u_θ

Transmission d'un écoulement tourbillonnant 3D ($u_\theta = f(x, y, z, t)$)

Un écoulement pleinement en rotation dans le tube est transmis d'une région 3D vers une région 2D axisymétrique (voir Figure 6.12). u_θ est fonction de la position dans le domaine et du temps. À l'entrée, on impose la vitesse ($u_r = 0.0,\ u_\theta = r \times \alpha,\ u_z = 30.0$) (avec, r le rayon et le coefficient constant $\alpha = 100.0$) pour créer un écoulement en rotation dans le tube. L'écoulement est en rotation à une vitesse dépendante du rayon, en conservant $r \times u_\theta = cst$. Ainsi on peut vérifier la déviation entre les champs de vitesse 3D et 2D axisymétrique. Le maillage n° 1-b est employé.

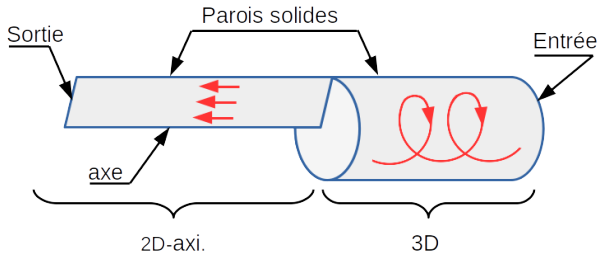


Figure 6.12 Configuration pour le transport de u_θ du domaine 3D au domaine 2D axisymétrique

Le tube est de rayon $R = 0.2\ m$ et de longueur $L = 0.6\ m$. La figure 6.13 montre la solution quasi-stationnaire, les résultats sont filtrés pour des valeurs de $u_\theta \approx 10.0\ m.s^{-1} \pm 0.01\ m.s^{-1}$.

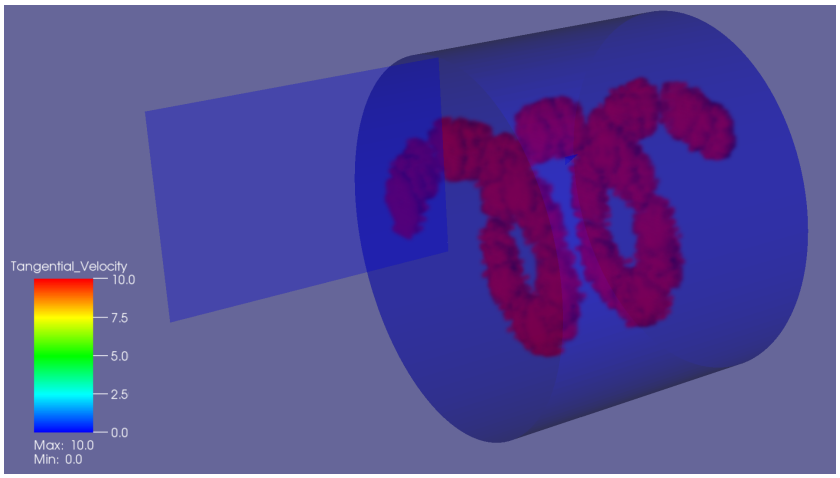


Figure 6.13 Solution (quasi-stationnaire) pour le transport de u_θ du domaine 3D au domaine 2D axisymétrique . Filtrage et affichage de la solution pour $r \times \alpha = u_\theta \approx 10.0\ m.s^{-1} \pm 0.01\ m.s^{-1}$ (La solution 2D est masquée pour améliorer la visibilité sur la figure)

La vérification du transfert de l'information durant la traversée du couplage est faite en comparant la dispersion spatiale de la quantité $\frac{1}{r} \times u_\theta = \alpha$ pour différents rayons à la fois dans les parties 2D axisymétrique et 3D. u_θ doit varier linéairement avec le rayon de 0.0 $m.s^{-1}$ à 20.0 $m.s^{-1}$, la comparaison est affichée sur la figure (6.14) où u_θ est dessinée pour trois rayons.

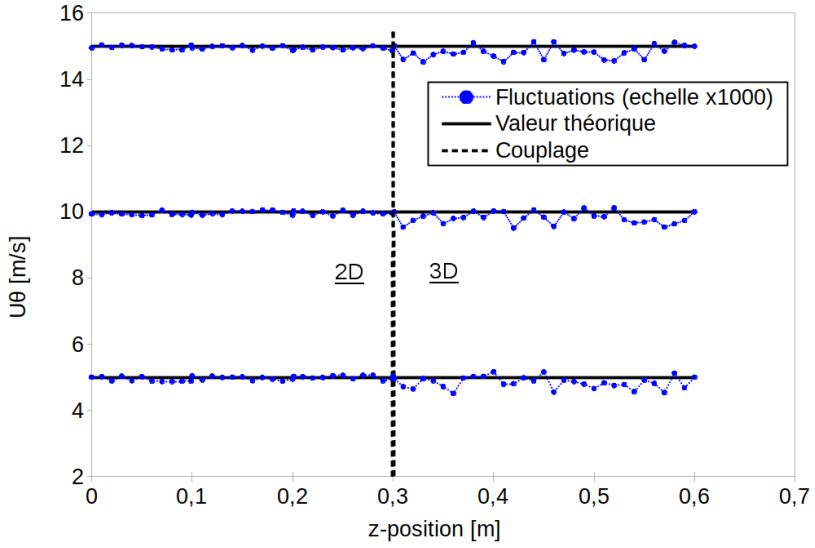


Figure 6.14 Écart de u_θ versus sa valeur théorique (variable avec le rayon, 3 sont tracés)

Les résultats du transfert, du domaine 3D au domaine 2D axisymétrique, s'accordent très bien avec la solution attendue, le couplage fonctionne bien également lorsque l'écoulement provient de la région 3D.

Sur la figure (6.14), on observe plus de fluctuations de u_θ dans le domaine 3D que dans le domaine 2D. Le maillage 2D, de type cartésien, donne des résultats plus stables que le maillage 3D composé de tétraèdres, plus sensibles aux erreurs de calculs en virgule flottante (projection des vitesses sur la normale $\vec{n}$, calcul des volumes, des surfaces, erreurs d'arrondi...). L'intensité des fluctuations est plus importante pour les maillages non-structurés que cartésiens et augmente lorsque la taille des volumes de contrôle grandit. Les fluctuations ne sont pas dues à la présence du couplage et sont la conséquence de phénomènes numériques "*normaux*" que nous avons observés avec des domaines de calcul uniquement 3D.

Transmission d'une onde de choc ($u_\theta = 0$)

Le problème du tube à choc permet d'analyser le comportement de la méthode de couplage lors d'écoulements compressibles et pour la transmission d'ondes de choc. La configuration

est indiquée sur la figure (6.15). La longueur totale du domaine est de 40.0 mm, avec la position du couplage située à $z = 20.0$ mm, tandis que le rayon du tube est de 0.5 mm. Le maillage n° 2 a été utilisé.

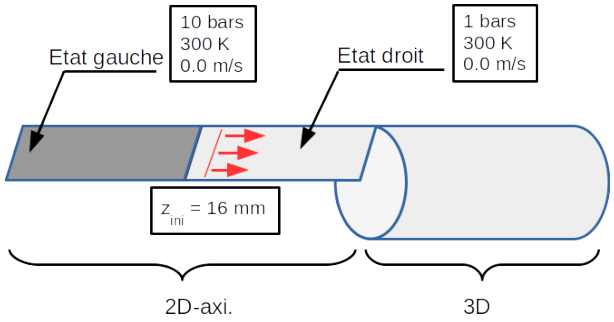


Figure 6.15 Configuration du tube à choc - Une onde de choc se déplaçant vers la droite traverse le couplage. Pour la traversée d'ondes de raréfaction, les états initiaux gauche et droite du tube à choc doivent être inversés

La séparation des états gauche et droite du choc est positionnée à $z = 16.0$ mm, ce qui signifie qu'une onde de choc se propage de cette position dans une direction dépendante de la position relative de la basse/haute pression. Une autre onde se dirige de cette position dans la direction opposée, l'onde de raréfaction (décompression).

On présente seulement les résultats pour un écoulement de fluide du 2D axisymétrique vers le 3D (l'inverse donnant des résultats totalement similaires, donc non présentés ici). Les résultats numériques sont comparés à la solution analytique.

Traversée d'une onde de choc ($u_\theta = 0$)

La figure 6.16 présente la distribution de masse volumique le long du tube. Les résultats concordent très bien avec la solution analytique. De plus, la différence des résultats est très difficilement discernable d'une solution couplée à une solution non-couplée.

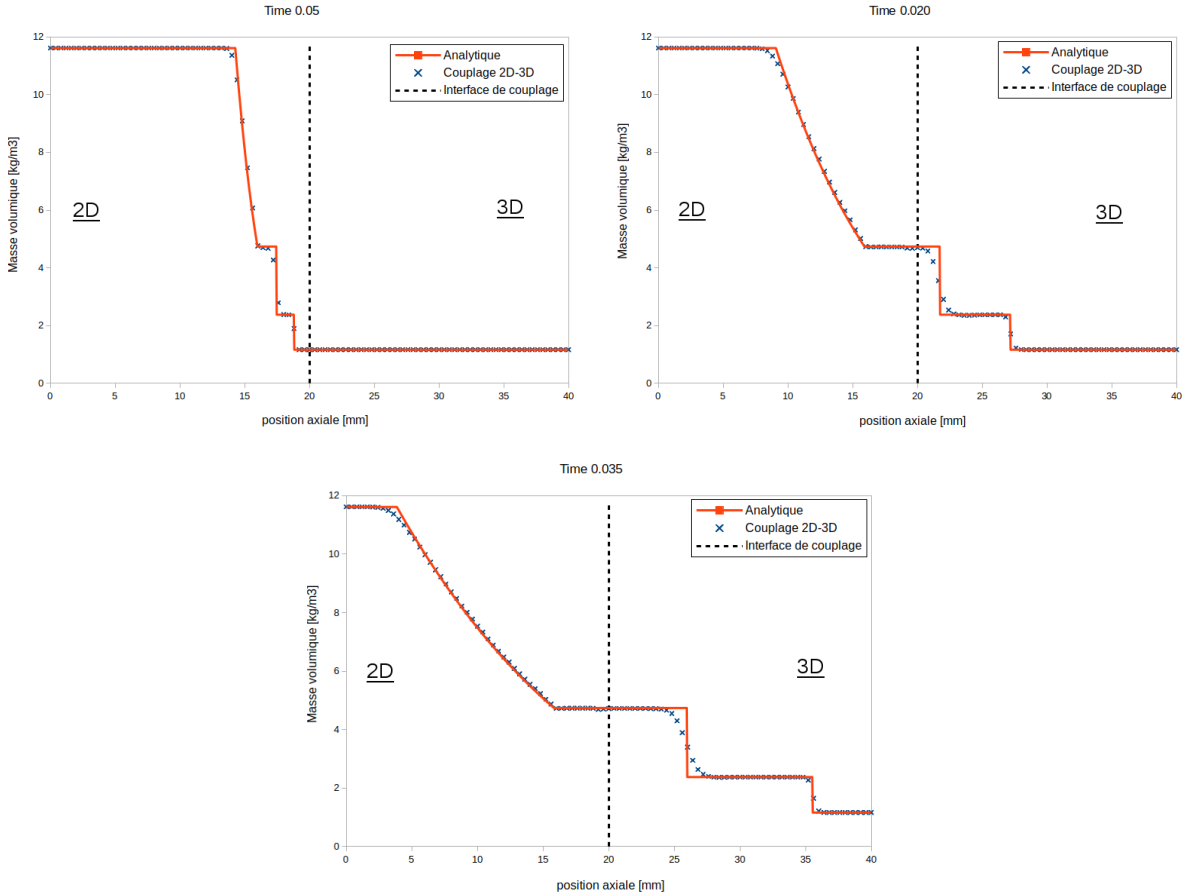


Figure 6.16 Traversée d'une onde de choc : masse volumique

Malgré la force importante du choc ($\frac{P_{2D}}{P_{3D}} = 10.0\ bars$), il est très bien capturé et transmis à travers l'interface de couplage. Un léger et négligeable effet est perceptible sur le tracé de la pression (Figure (6.17)), dans la partie 2D du domaine. Le front du choc n'est pas perturbé par la traversée.

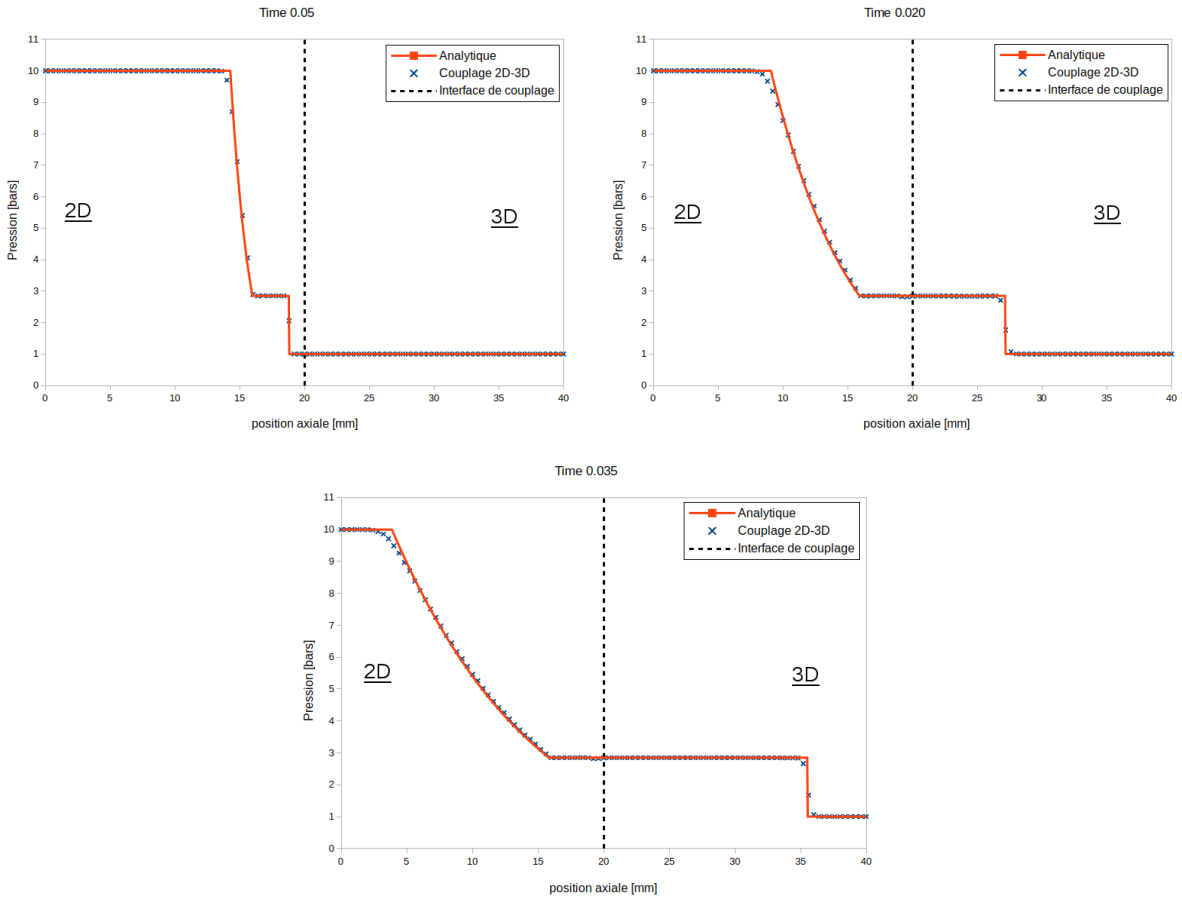


Figure 6.17 Traversée d'une onde de choc : pression

La vitesse est la plus sensible des quantités, telle qu'illustrée sur la figure (6.18) où est tracé le nombre de Mach à différents temps. On peut voir un léger sursaut dans la partie 2D axisymétrique du domaine juste avant le couplage. Le changement local du maillage en taille, en régularité et le changement de la section de passage d'un domaine à l'autre peuvent expliquer ce phénomène. Néanmoins, ce sursaut est faible, l'erreur locale est inférieure à 2%.

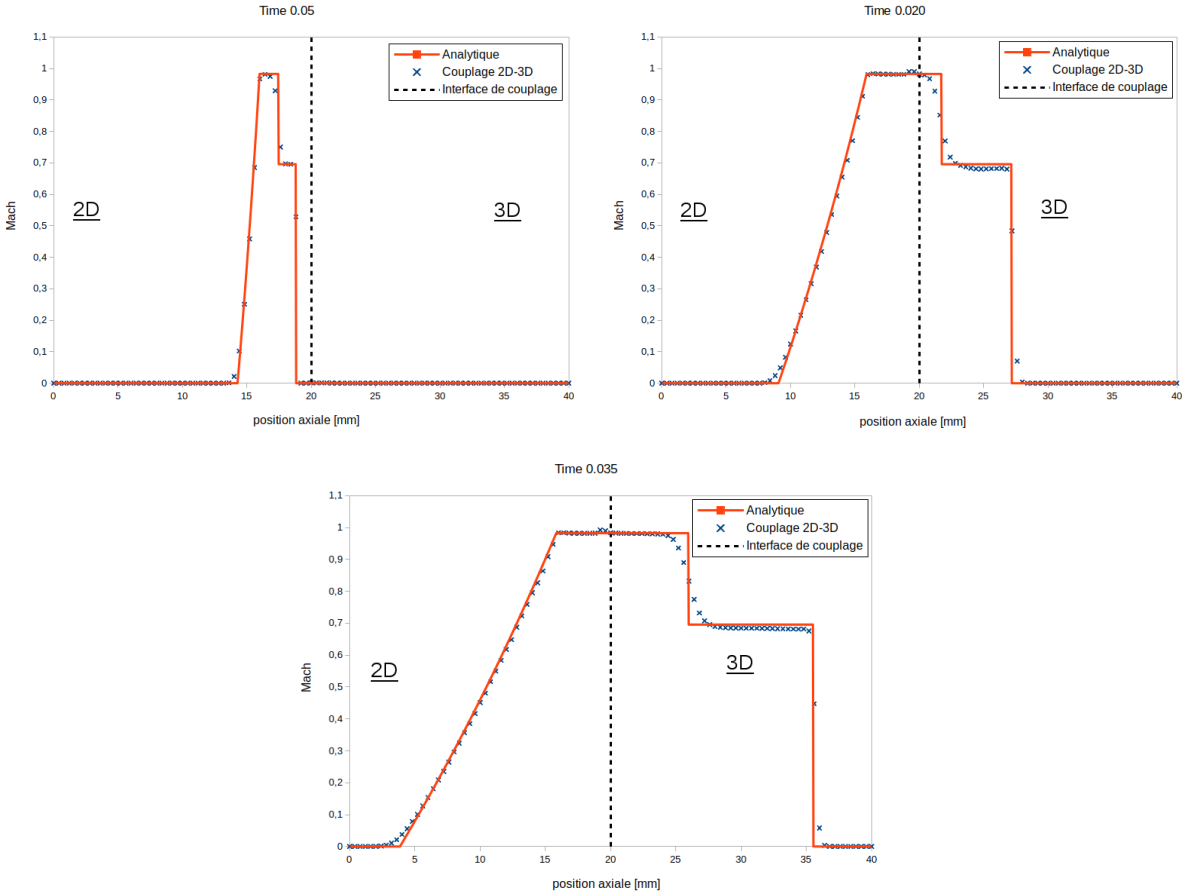


Figure 6.18 Traversée d'une onde de choc : Mach

Transmission d'une onde de raréfaction ($u_\theta = 0$)

Les ondes de raréfaction, aussi connues sous le nom de *détente de Prandtl-Meyer*, sont des ondes de détente se propageant dans la région de plus haute pression. La variation de pression est lisse et la transformation est isentropique. La conservation de l'entropie à travers le couplage peut ainsi être vérifiée avec ce cas test. Les figures (6.19), (6.20) et (6.21) présentent les résultats pour la distribution de masse volumique, de la pression et du nombre de Mach respectivement.

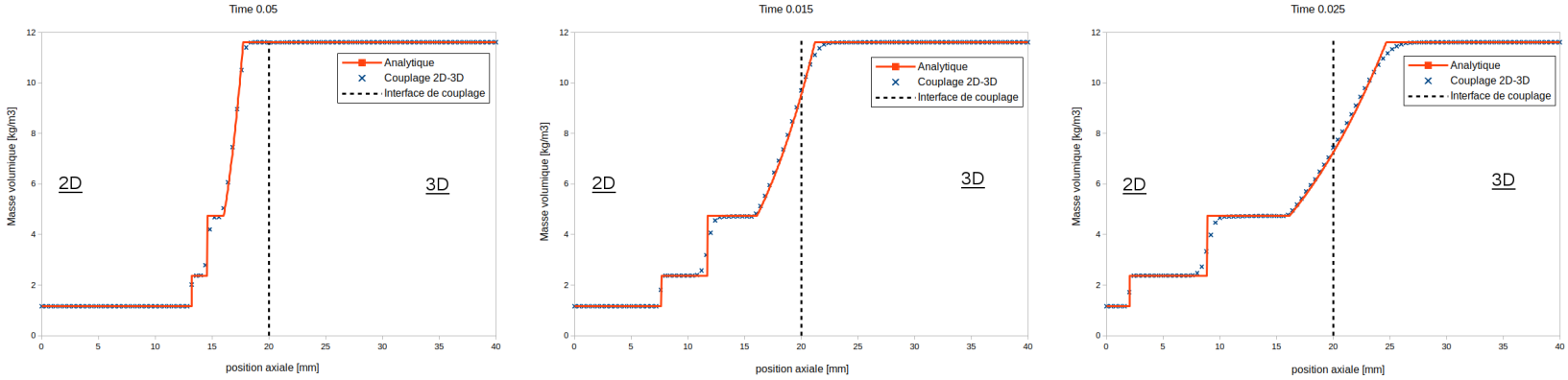


Figure 6.19 Traversée d'une onde de raréfaction : masse volumique

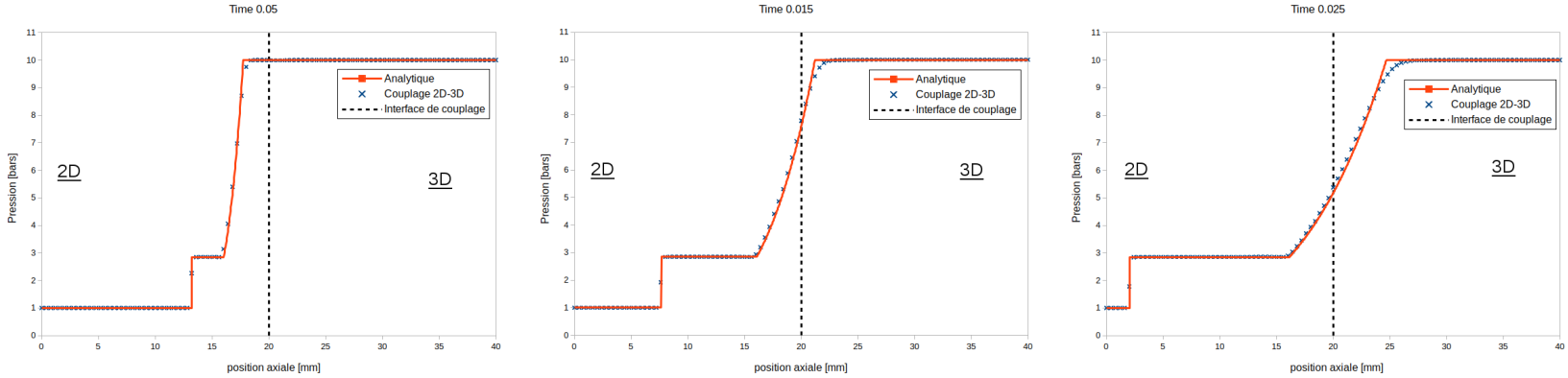


Figure 6.20 Traversée d'une onde de raréfaction : pression

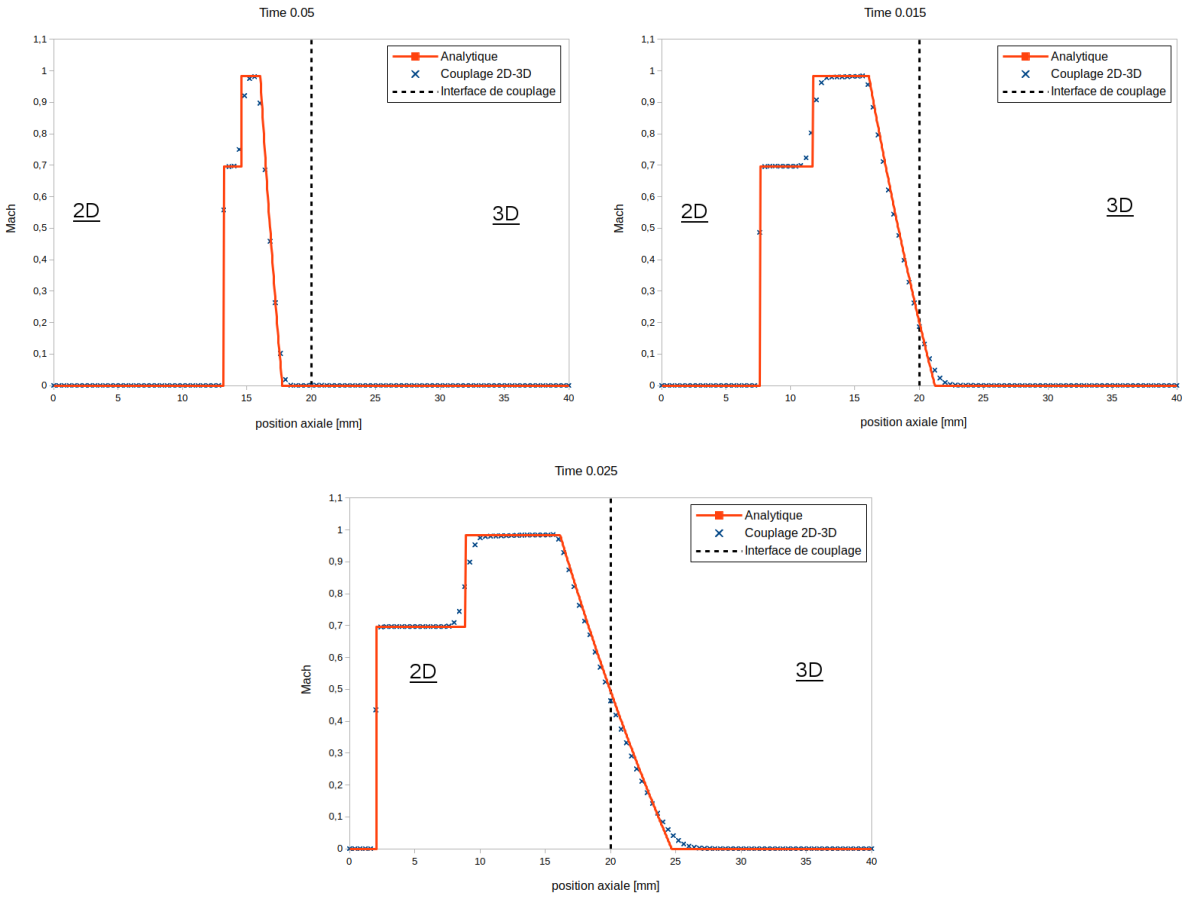


Figure 6.21 Traversée d'une onde de raréfaction : Mach

Ces résultats numériques sont pratiquement identiques aux résultats non-couplés (complètement 2D ou complètement 3D) et valident la méthode de couplage. La présence de l'interface de couplage est quasiment imperceptible.

6.5.2 Couplage radial

Tous les cas test précédents ont permis de vérifier le couplage **axial**, la vitesse u_r étant considérée nulle. Dans les prochains cas test, un couplage de type **radial** est mis en place et la vitesse u_r est variable.

Écoulement stationnaire dans une tuyère ($u_\theta = 0$)

Un écoulement compressible radial et stationnaire dans une géométrie 2D axisymétrique (Figure 6.22) avec des conditions d'entrée/sortie à rayons constants donnés (ou 3D en forme d'anneau) est équivalent à un écoulement dans une tuyère. La position de la région 3D est

du côté intérieur de la géométrie, le positionnement relatif inverse a également été étudié et produit des solutions numériques semblables qui ne seront pas présentées. Une solution analytique est disponible et sera utilisée pour analyser les capacités de transmission d'écoulement compressible à travers l'interface de couplage, cette fois-ci dans une position radiale. u_θ n'est pas imposée et reste libre d'évoluer (mais doit rester proche de zéro aux erreurs d'arrondis près). Les simulations sont réalisées avec le maillage n^o 3.

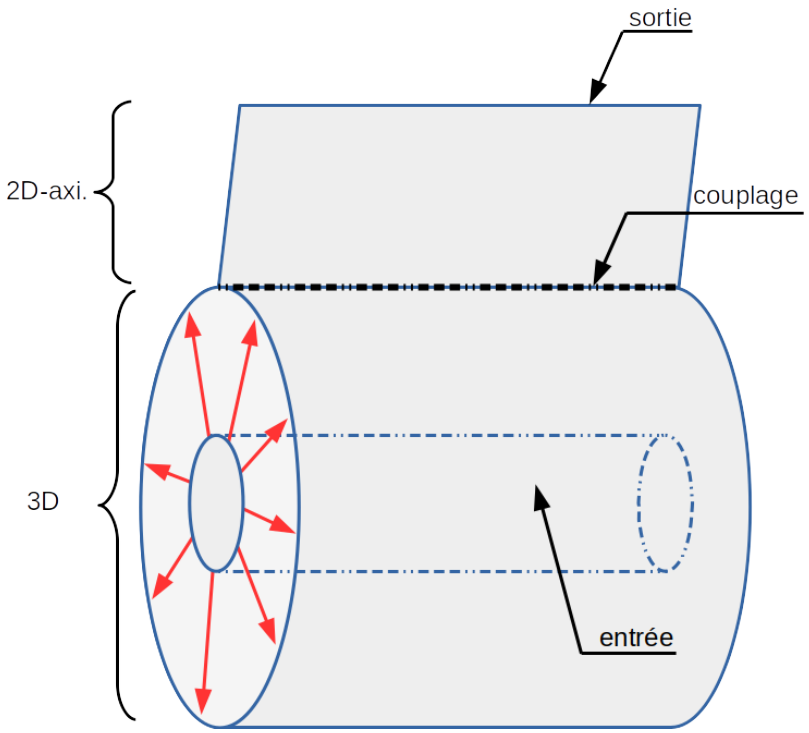


Figure 6.22 Configuration du cas test d'écoulement radial

La condition d'entrée est positionnée à un rayon $Rmin = 1.0\ mm$, l'interface de couplage à $r = 1.5\ mm$ et le rayon de sortie est $Rmax = 3.0\ mm$. La condition aux limites d'entrée est réglée à la pression totale $P_0 = 1.0\ bar$ et la température totale $T_0 = 300.0\ K$, tandis que la condition aux limites de sortie est configurée à la pression statique $P_s = 0.9785\ bar$.
Les simulations numériques sont comparées avec la solution analytique, la figure (6.23) présente la répartition de pression le long de la position radiale et la figure (6.24) montre le nombre de Mach.

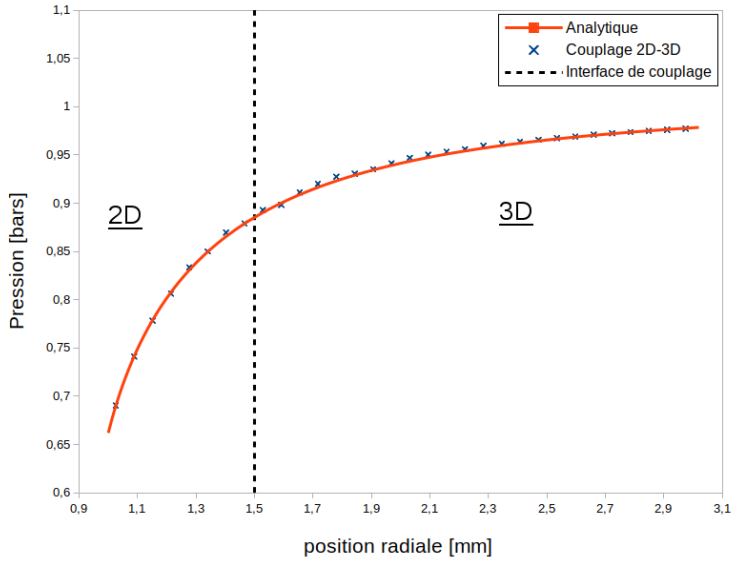


Figure 6.23 Écoulement radial : pression

De petites oscillations sont perceptibles sur la courbe de pression, près du couplage aussi bien dans la région 2D axisymétrique que dans la région 3D. Néanmoins, la méthode de couplage fonctionne bien et la simulation est proche de la solution analytique.

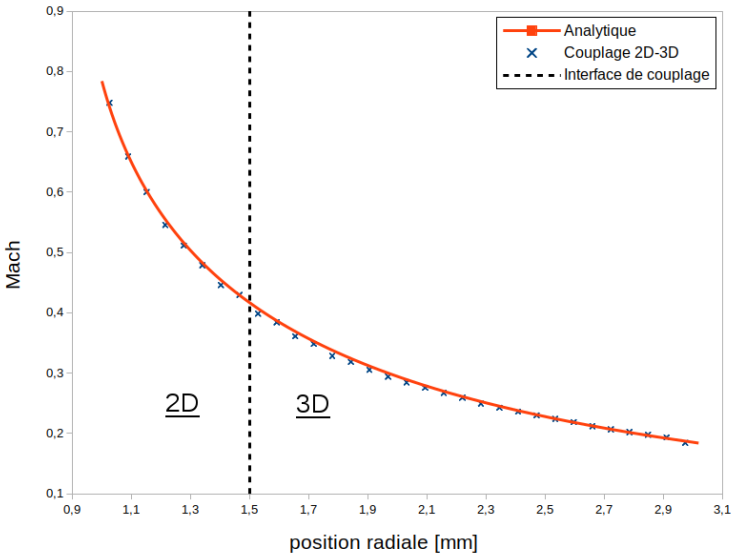


Figure 6.24 Écoulement radial : Mach

Le même type d'oscillations apparaît avec l'évolution du nombre de Mach le long de la position radiale, mais elles restent aussi faibles.

Le comportement de la méthode de couplage est en très bon accord avec la solution analytique (aucune vitesse u_θ n'est transportée).

Transport radial d'un tourbillon ($u_\theta \neq 0.0\ m.s^{-1}$)

Le transport de u_θ dans un écoulement équivalent à une tuyère est réalisé, ainsi, un calcul quasi-stationnaire est considéré. La configuration est présentée à la figure (6.25). Le maillage n° 3 est employé ici.

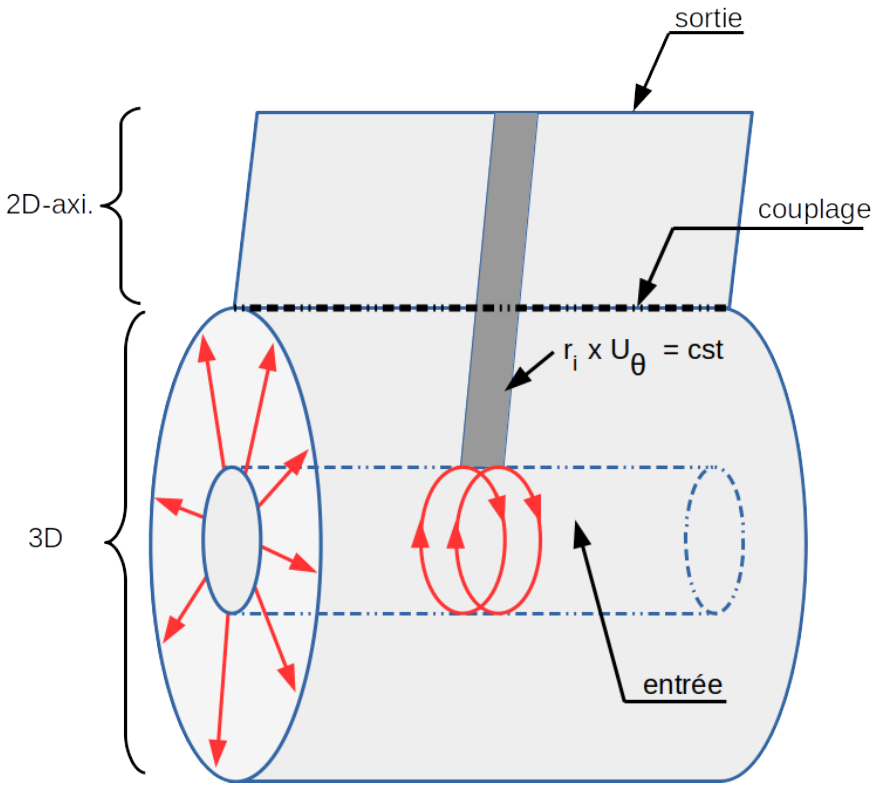


Figure 6.25 Configuration pour le transport radial de u_θ

On impose $r \times u_\theta = 10.0\ m^2.s^{-1}$ au rayon d'entrée.

La vérification de la transmission du couplage pour cet écoulement radial a été étudiée et la figure 6.26 montre l'évolution du ratio $r \times u_\theta$ le long de la position radiale, ratio devant demeurer constant. Le profil est préservé avec la traversée et le transport. La méthode de couplage fonctionne également très bien pour les couplages radiaux.

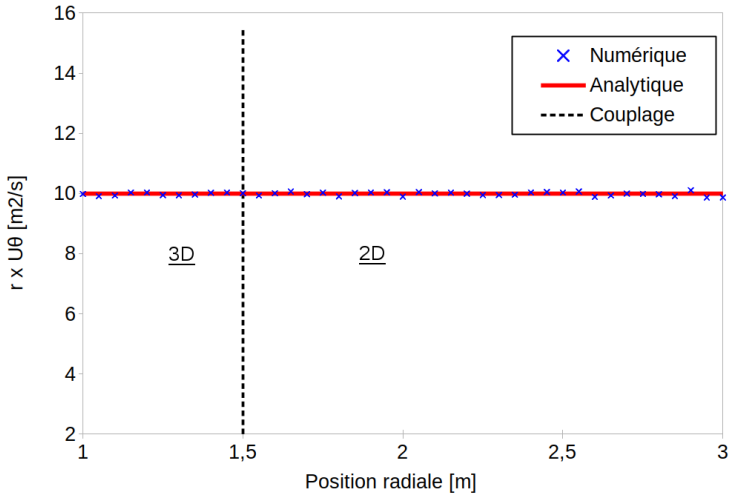


Figure 6.26 Déviation autour de $r \times u_\theta = cst = 10.0\ m^{-2}.s^{-1}$

6.6 Estimation du gain de performances pour les disjoncteurs

Afin d'estimer le gain de vitesse d'exécution d'une solution hybride (2D axisymétrique-3D) par rapport à une solution 3D complète, des métriques de calcul sont nécessaires. La première étape consiste à déterminer le ratio du volume des régions 3D sur le volume géométrique total. Avec les régions d'intérêts pour le 3D identifiées en rouge à la figure 6.27, on estime que ce ratio de volume est de 10 à 20%. En supposant un gain de vitesse linéaire, le temps nécessaire à la résolution hybride 2D axisymétrique 3D (≈1 million d'éléments) est réduit d'un facteur 5x à 10x par rapport à un calcul complètement 3D.

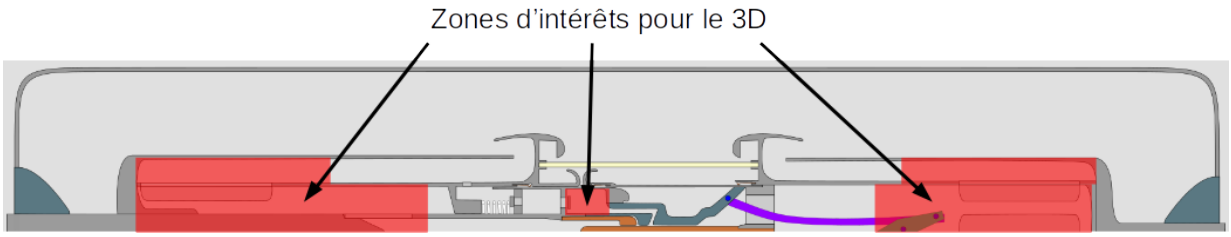


Figure 6.27 Zones identifiées d'intérêts pour les calculs 3D

La deuxième étape concerne la génération et la maintenance du maillage. Lors de l'utilisation d'une méthode hybride, la complexité du maillage 3D est diminuée car la géométrie est plus simple dans les régions souhaitées. Cela permet de réduire la durée nécessaire à la génération du maillage initial qui est une tâche approchant plusieurs semaines pour une géométrie de disjoncteur. La gestion d'un maillage mobile adaptatif 3D est également très coûteuse et

une robustesse suffisante des algorithmes est extrêmement difficile à obtenir (aucun logiciel commercial n'en est capable à l'heure actuelle et il s'agit d'un défi scientifique et technique non encore résolu). Dans le cas hybride, les zones d'intérêts 3D identifiées concernent des régions où le maillage est fixe ou simplement translaté uniformément, ce qui annule ou minimise le coût de maintien du maillage 3D au fil des itérations. Une économie supplémentaire de temps de traitement et d'exécution est sauvée (facteur estimé supérieur à 2x).

La troisième composante concerne le cluster de calcul. En effet, la quantité de processeurs et de mémoire DRAM requise impacte les temps de résolution. Avec une taille de problème résolu plus importante, moins de calculs peuvent être réalisés en parallèle sur le même cluster. Un cas test semi-industriel représentatif a été réalisé (Figure 6.28) pour estimer quelques métriques de calcul qui sont résumées dans le tableau 6.1. Un écoulement de gaz chaud (6000 K) entrant à grande vitesse ($M = 1.4$) dans un volume équivalent à un volume thermique de disjoncteur est simulé avec le maillage n^o 4.

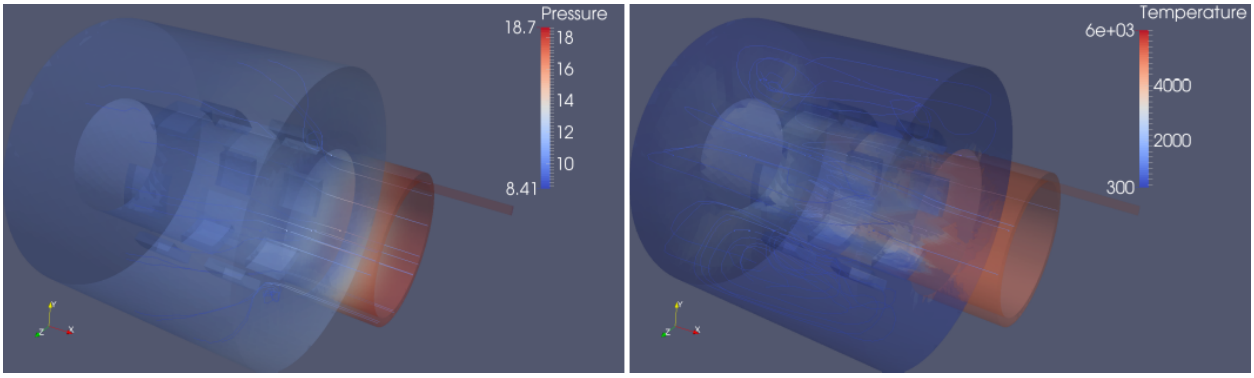


Figure 6.28 Résultats illustratifs sur le cas semi-industriel

Tableau 6.1 Récapitulatif de métriques de calcul

thèmes	2D	2D-3D	3D complet
Usage DRAM (estimé)	$\approx 0.1GB$	$< 16GB$	$192 - 256GB$
nombre de nœuds / cœurs (estimés)	1 / < 32	$1 - 2$ / < 64	$4 - 8$ / > 128
temps (estimé)	1	$20 - 40$	> 200

D'autres validations seront nécessaires pour le couplage d'arc et l'intégration avec MC^3 et constitueront des travaux futurs.

6.7 Extension de l'architecture logicielle

L'architecture logicielle présentée à la section 5, se prête bien aux extensions futures. Pour preuve, les fonctionnalités du solveur sont facilement enrichies avec la méthode de couplage, le nouveau diagramme de classes correspondant à l'outil étendu est présenté à la figure 6.29 où les classes vertes ont été ajoutées au diagramme 5.4. En rouge, apparaissent les points de connexion avec le solveur précédent, purement 2D axisymétrique.

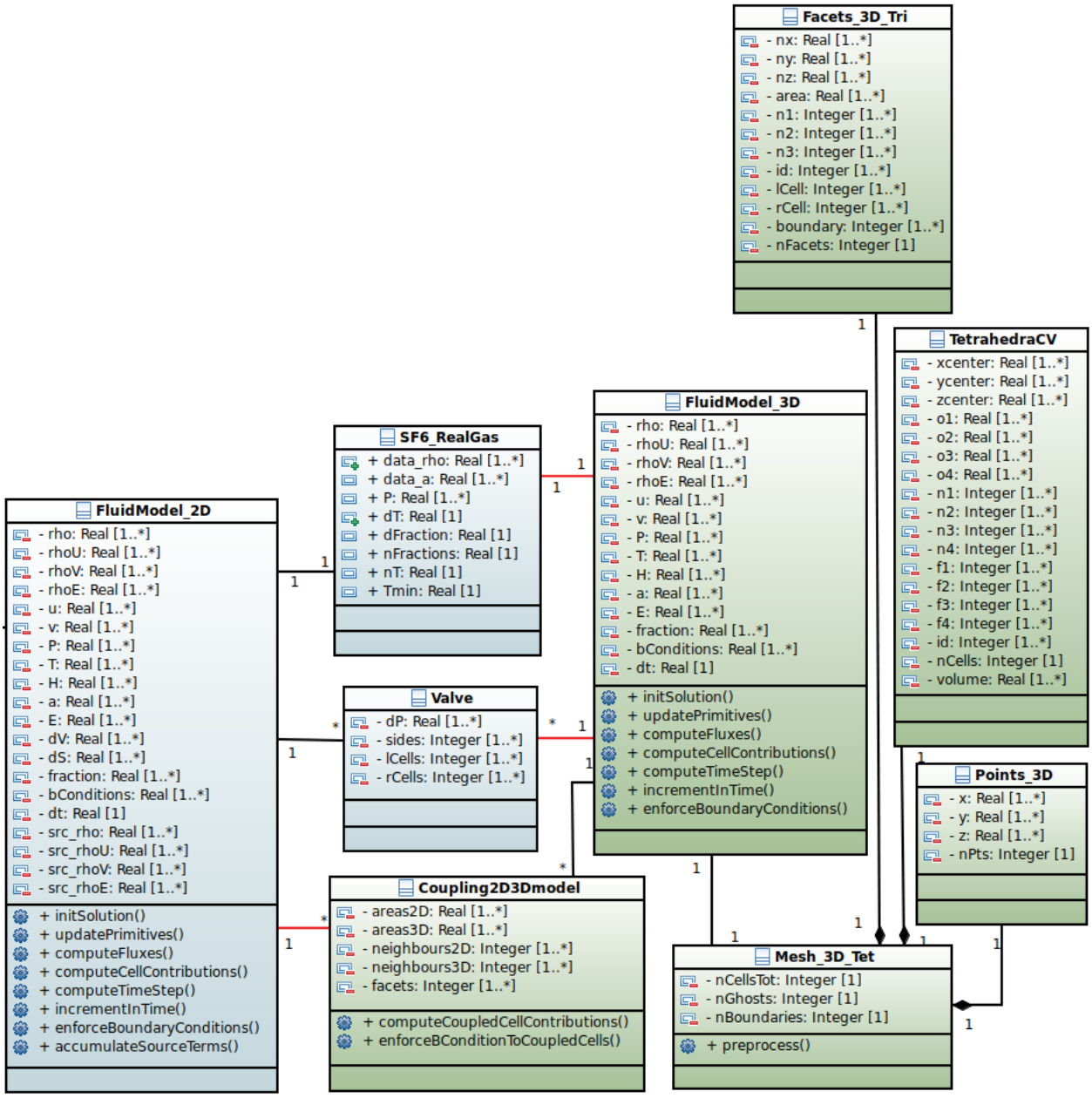


Figure 6.29 Diagramme de classes (partiel) avec l'extension du couplage 2D axisymétrique-3D

CHAPITRE 7 CONCLUSION

Ce chapitre présente la conclusion de cette recherche. Tout d'abord, une synthèse des travaux sera réalisée. Les objectifs atteints et les contributions scientifiques qui en découlent seront alors décrits. Ensuite, les limitations des solutions proposées dans ce document seront présentées dans une deuxième section. Finalement, la dernière section introduira les améliorations futures et les champs de recherche pouvant être poursuivis à l'issue de cette recherche.

7.1 Synthèse des travaux

Depuis le début des années 1990, la simulation numérique est un domaine scientifique en plein essor. Le groupe GRMIAO de l'École Polytechnique de Montréal, fut un des précurseurs dans le développement d'outils de simulation numérique assistée par ordinateur et notamment dans le domaine de la simulation numérique des disjoncteurs à haute-tension à l'origine de l'outil MC^3, aujourd'hui utilisé par l'industriel GE Grid Solutions pour l'étude et la conception des appareils de coupure d'arcs électriques. Cependant, au fil des années, beaucoup d'intervenants, dont de nombreux étudiants, ont contribué à l'écriture de MC^3 qui a dérivé petit à petit vers un code spaghetti. La faisabilité et la qualité de futures extensions étaient devenues trop basses. De plus, les algorithmes implémentés se sont révélés peu efficaces en vitesse d'exécution et difficilement parallélisables. De ces constats et de la volonté d'extension de l'outil, le besoin de moderniser MC^3 s'est fait ressentir. C'est dans cet axe de recherche que s'inscrit cette thèse de doctorat. Dans un but de modernisation de MC^3, en améliorant à la fois ses performances en temps de calcul et sa prédiction des écoulements, la thèse a visé les objectifs généraux suivants :

1. **Choisir les modèles pertinents :**
   Le premier objectif était de confirmer ou infirmer les choix des modèles en place dans MC^3. Cette tâche s'appuyait en partie sur la revue de la littérature.
2. **Optimiser les modèles avec le support de l'analyse de MC^3 :**
   Le second objectif était d'analyser MC^3 de manière approfondie pour ensuite proposer des stratégies d'amélioration bénéfiques à la vitesse d'exécution et/ou à la précision des calculs. Les points d'analyse ont porté sur les axes suivants : des structures de données pour le maillage et les quantités physiques ; des formats matriciels, des solveurs linéaires ; des algorithmes pour exposer au maximum le parallélisme et la scalabilité ; du style d'écriture du code source pour favoriser l'interprétation par le compilateur et donc l'optimisation de l'exécutable final.

3. **Proposer une architecture adéquate :**
   Les différents modèles physiques doivent être couplés de façon adaptée et exposer le parallélisme de tâches. Les opérations de maintenance contribuent au coût financier global du logiciel avec une portion supérieure à 95 % (Pigoski [93]). La performance en temps de calcul, la lisibilité du code source et la facilitation des évolutions futures et de la maintenance sont des points qui peuvent se révéler contradictoires. Le troisième objectif a produit une architecture conçue pour présenter un équilibre entre ces contraintes.

4. **Améliorer la prédiction des simulations :**
   La précision des simulations numériques peut être améliorée par l'ajout des calculs de mécanique des fluides tri-dimensionnels, cela a constitué notre quatrième objectif. Il s'agissait du premier point d'amélioration à prendre en compte car cela influence directement la conception des appareils. Cependant, résoudre les écoulements fluides en 3D pour une géométrie complète s'avère à ce jour inefficace et non-pertinent. À l'inverse, combiner des zones de calculs 3D en compléments de zones 2D axi-symétriques s'avère plus judicieux. Nous avons donc développé une méthode de couplage entre les régions de différentes dimensions.

Premièrement, la revue de littérature a démontré une base solide au niveau des choix des modèles physiques implémentés et de leurs modélisations numériques. MC^3 était donc plutôt à jour vis-à-vis de l'état de l'art. Cependant, concernant la mécanique des fluides, le schéma original de Roe [106] pourrait être modernisé en utilisant plusieurs extensions. Le schéma implémenté dans MC^3 comprend les extensions aux propriétés de gaz réels, proposées par Glaister [33] et Godin et al. [35]. Les propriétés primitives telles que la pression et la température sont mises à jour en utilisant une table de données thermodynamiques de gaz réel. Le schéma est alors partiellement adapté pour les gaz réels mais fondamentalement il reste quasiment identique à celui utilisé en gaz parfait. Pour le rendre pleinement adapté aux gaz réels, on a intégré la correction proposée par Arabi et al. [7] qui permet de renforcer le respect de la 3ème loi de Roe concernant l'équation d'énergie.

Deuxièmement, l'étape suivante de ce travail de recherche, a été d'optimiser l'efficacité d'exécution du programme, en réduisant le temps nécessaire à la réalisation des calculs. Pour cela, l'analyse de MC^3 à l'aide d'outils de profilage et de métriques CPU a été une source importante d'informations. Une connaissance fine des détails de fonctionnement des disjoncteurs haute-tension et des modèles numériques employés dans leur modélisation, combinés à une connaissance approfondie du fonctionnement d'un point de vue matériel des processeurs et des cartes accélératrices de calcul (GPU, Xeon Phi), ont permis de proposer des améliorations

apportant des gains significatifs de vitesse d'exécution. Les propositions d'améliorations qui ont été implémentées sont de natures fonctionnelles, structurelles, algorithmiques et parallèles et ont mené à des changements importants, si bien que la réécriture complète de l'outil a été entreprise. Pour cela, le langage C++ a été choisi, afin de bénéficier de la flexibilité et des hautes performances de ce langage et d'un environnement de développement moderne et efficace. La méthodologie générale de modernisation de MC^3 a été la suivante :

I. Créer la suite de tests de non-régression et de validation utilisée pour toutes les étapes d'optimisation.

II. Nettoyer le code désuet.

III. Isoler les physiques.

IV. Identifier les points critiques à accélérer.

V. Accélérer les physiques et les points critiques un à un.

VI. Intégrer les points accélérés et réviser l'architecture (troisième objectif de la thèse).

VII. Améliorer la prédiction en ajoutant un modèle fluide 3D partiel et un modèle de couplage 2D axisymétrique-3D (quatrième objectif de la thèse).

Le point V a été développé selon les **six phases consécutives d'optimisation** :

1. Révision algorithmique et de modélisation.
2. Adaptation au problème résolu : spécialisation pour les disjoncteurs haute-tension.
3. Optimisation arithmétique, du CPI et du parallélisme d'instruction (ILP).
4. Optimisation du parallélisme de vecteur (VLP).
5. Optimisation du parallélisme de tâches (TLP).
6. Optimisation de l'efficacité mémoire.

Chacune des six phases est étroitement liée aux autres, et participe à améliorer les performances des cinq autres. Toutes les physiques entrant en jeu dans MC^3 ont obtenu de fortes accélérations qui s'échelonnent entre trente et six-cents fois environ. Cependant, en conséquence des deux premières phases d'optimisation, les facteurs d'accélération sont dépendants du problème : la géométrie et les choix pratiques de définition des conditions de simulation ont un impact sur l'accélération globale.

Troisièmement, l'architecture logicielle devait être révisée car elle était inadaptée d'une part aux calculs multiphysiques et d'autre part au parallélisme et aux architectures du matériel moderne. Les performances finales du logiciel, sont un des nombreux points à prendre en

compte, l'application doit aussi être facile à prendre en main, suffisamment souple et évolutive pour faciliter les mises à jour futures. La fiabilité et la compatibilité avec les outils et matériels existants sont d'autres points à considérer. Afin de concevoir une bonne architecture, donc un bon logiciel, il a été vital d'avoir une vision d'ensemble approfondie de toutes les fonctionnalités, modules, dépendances, et de *visualiser* conceptuellement le projet dans sa globalité. La nouvelle architecture a permis de mettre en connexion les différents modules constitutifs du solveur multiphysiques et la structure proposée permet un équilibre entre les compromis de performance en vitesse d'exécution, de facilité de maintenance et d'évolutivité de l'outil. La compréhension et la lisibilité du code source ont grandement été améliorées par l'architecture proposée et le style d'implémentation. La combinaison de la nouvelle architecture et du passage d'un langage procédural (FORTRAN 77) à un langage orienté objet (C++) a aussi permis de mieux structurer le code source (figure 5.5). De plus, la décomposition en terme de tâches exécutées en parallèle a été réalisée à différents niveaux d'imbrication. En effet, pendant le même pas de temps, toutes les physiques ont été rendues indépendantes et les appels et les résolutions sont réalisés avec des tâches parallèles non-bloquantes imbriquées. Les résolutions ne nécessitent pas de synchronisation, excepté avant l'intégration temporelle et à la fin d'une itération. Le scheduler de la bibliothèque TBB, qui est utilisé pour le parallélisme, se charge de mapper les tâches sur les threads dynamiquement selon les contraintes du matériel (proximité en cache, affinité au fil des itérations, etc). Ainsi, la vitesse d'exécution et la scalabilité sont élevées (figure 5.10) même lorsque la taille du problème reste contenue et les performances continueront de s'améliorer avec les générations futures de CPU sans efforts particulièrement importants. Les vitesses d'exécution relevées sur quelques cas d'application avec des solutions initiales de disjoncteur, permettent d'estimer l'accélération globale du nouvel outil par rapport à la version de référence de MC^3 à un facteur compris entre trente et cent fois.

Quatrièmement, la précision des simulations numériques pourra être améliorée par l'ajout des calculs de mécanique des fluides tri-dimensionnels. En effet, la résolution en 3D a le potentiel d'améliorer la fidélité des résultats numériques versus les mesures d'essais et d'apporter de nouvelles perspectives d'amélioration lors du design des appareils de coupure. Cependant, comme on cherche à optimiser le temps de résolution, l'idée d'un couplage de la mécanique des fluides entre des régions 2D axisymétriques et des régions 3D a émergé. Ainsi, des calculs fluidiques dans des régions 3D, complètent les calculs multiphysiques d'arc réalisés dans d'autres zones 2D axisymétriques. Afin d'assurer l'échange d'information entre les régions de différentes dimensions, une nouvelle méthode de couplage a été développée dans le cadre de cette thèse. La méthode proposée permet, non seulement, d'autoriser le transfert d'information de manière bi-directionnelle, c'est-à-dire aussi bien pour des directions d'écoulement de

la partie 2D axisymétrique vers la partie 3D, et inversement, mais elle fonctionne aussi pour des écoulements subsoniques ou supersoniques à travers l'interface. De plus, contrairement à certaines méthodes de couplage, telles que les techniques à base de *General Grid Interface* (GGI), telle que l'interface *mixing plane* notamment intégrée à Fluent [6], la nouvelle méthode développée respecte la conservation de la masse, de la quantité de mouvement et de l'énergie. Lors des validations, elle a démontré de très bons résultats aussi bien dans le cadre de couplages axiaux que radiaux : les résultats avec et sans couplage sont difficilement différentiables. Une solution hybride permet une réduction du temps de calcul par rapport à une solution 3D complète d'un facteur estimé entre cinq et dix fois. L'extension de l'outil a été très facile grâce à la nouvelle architecture préalablement développée lors de la réalisation du troisième objectif.

Au final, l'amélioration de l'outil numérique s'est faite tant d'un point de vue précision et fidélité des calculs, que d'un point de vue de la vitesse d'exécution grandement améliorée. Ainsi, le nouvel ensemble permet de réaliser des calculs 3D (là où nécessaires géométriquement) à la vitesse des calculs 2D axi-symétriques anciennement proposés par l'ancien outil de calcul. Parallèlement, la nouvelle architecture logicielle apporte des améliorations significatives en termes de qualité logicielle.

7.2 Limitations des solutions proposées

Les améliorations de vitesse d'exécution sont en partie obtenues grâce à un style de programmation, des techniques avancées issues du savoir-faire de l'auteur et de la communauté du HPC, et du langage assembleur. Ceci rend, certaines portions de code beaucoup moins accessibles aux développeurs non-spécialisés ou néophytes. De plus, les modifications qui pourraient être apportées par un nouveau développeur peuvent potentiellement impacter négativement les performances en vitesse d'exécution. Confier la tâche de développer ou d'intégrer de nouveaux modules à des étudiants semble une tâche à éviter à moins d'une très forte expérience en programmation informatique. Dans le cas contraire, la maintenabilité et les performances pourraient s'amenuiser.

Au niveau de la nouvelle architecture logicielle, une partie de la solution proposée repose sur une librairie externe (TBB). La gestion du planificateur interne de TBB comporte des paramètres inaccessibles depuis les interfaces de programmation, c'est à la fois une force pour le programmeur, pour des questions de simplicité et d'accessibilité, mais c'est aussi une faiblesse car un contrôle plus précis pourrait permettre d'obtenir des gains supplémentaires en vitesse d'exécution. Cependant, cela reste une économie en temps de programmation et en maintenance et facilitera les migrations futures vers du nouveau matériel sans efforts

particuliers de migration.

La méthodologie de couplage 2D axisymétrique-3D et les solveurs de mécanique des fluides ont été développés pour les équations d'Euler. Cependant, pour les écoulements présents dans les chambres d'échappements, les volumes sont grands et la vitesse des écoulements chute d'un facteur important. La prise en considération des phénomènes de viscosité (équations de Navier-Stokes) est envisageable. De plus, la méthodologie de couplage telle que présentée dans cette thèse, a été développée avec une précision au premier ordre en espace. La méthode requiert le transport d'un scalaire supplémentaire dans le domaine 2D axisymétrique pour prendre en compte la vitesse azimutale, son coût reste cependant faible car c'est une méthode de transport convective pure. L'équation supplémentaire pour le transport de la vitesse azimutale doit aussi être intégrée temporellement. Un point limitant additionnel concerne les maillages 2D et 3D utilisés : les effets de filtrage fréquentiel à l'interface entre les maillages peuvent être présents si le changement de taille des éléments à l'interface est trop brutal. Ces effets peuvent notamment induire des réflexions d'ondes et/ou des solutions non-physiques dans les cas les plus extrêmes.

7.3 Améliorations futures

7.3.1 Schéma numérique appliqué à la mécanique des fluides

Le schéma de Roe s'est avéré très robuste lors de l'utilisation de MC^3 chez l'industriel, notamment lors des écoulements à nombres de Mach élevés (2 à 3 Mach) en présence de gaz à très haute température. En revanche, le schéma de Roe a été initialement conçu pour la résolution numérique des écoulements compressibles autour des profils d'ailes d'avions et pour des régimes de vitesse allant de transsoniques à hypersoniques. Dès lors que les vitesses d'écoulements sont plus faibles, le schéma numérique n'est pas adapté. Les termes acoustiques et convectifs sont différents de plusieurs ordres de grandeur et la résolution fournie par le schéma n'est plus représentative de l'écoulement réel qui se rapproche d'une solution incompressible. Des tentatives d'amélioration pour rendre le schéma de Roe adapté à toutes les vitesses d'écoulements ont été proposées par de nombreux auteurs tels que Rieper [104], Pelanti [90], Osswald *et al.* [85], Thornber *et al.* [119]. Rieper [104] propose de prendre en compte les régions de l'écoulement à bas nombre de Mach avec un simple facteur de mise à l'échelle basé sur le nombre de Mach. Des études supplémentaires en tant que travaux futurs seront requises pour choisir quelle(s) correction(s) appliquer. Dans ce travail, il n'a pas été souhaité de modifier trop profondément le schéma de Roe ni de changer de schéma numérique (par exemple, AUSM [65] pourrait être un remplaçant potentiel).

7.3.2 Intégration temporelle

Lors de la proposition de recherche initiale, il avait été envisagé d'employer une méthode d'intégration temporelle hybride, à la fois **im**plicite et **ex**plicite (**IMEX**). Cette classe de méthode est applicable à une variété de physiques ([88, 124, 127, 64, 52]) et du point de vue des disjoncteurs, la zone d'arc est portée à haute température et à haute vitesse d'écoulement, ce qui numériquement implique, pour des raisons de stabilité, des pas de temps très faibles ($\approx 10^{-5}\ ms$). Inversement, dans les régions les plus froides et/ou à faible vitesse d'écoulement, les pas de temps permis par les conditions de stabilité et la capture des phénomènes physiques sont plus importants de plusieurs ordres de grandeur. Cependant, ces régions sont soumises à la même incrémentation temporelle. Une méthode d'intégration temporelle implicite localement permet de s'affranchir de la contrainte de stabilité sur le pas de temps, au prix d'un coût de résolution plus élevé. Le but étant d'accélérer les calculs, en augmentant le pas de temps dans la zone d'arc, ce surcoût en temps de résolution doit être à minima compensé par cette augmentation. Comme on cherche aussi à capturer les chocs, le pas de temps implicite ne peut pas être trop élevé, sinon la diffusion numérique associée sera trop importante [13]. Finalement, appliquée aux disjoncteurs haute-tension, l'augmentation du pas de temps visée ne permet pas de compenser le surcoût de résolution. La méthode IMEX n'est donc pas rentable pour notre problème lorsque du gaz SF6 est utilisé. En revanche, dans le cas d'un changement de gaz de remplissage par du CO2, les propriétés de transport sont différentes, et une intégration temporelle moderne pourrait être bénéfique. Un travail de recherche pourrait être effectué pour étudier différentes approches d'intégration temporelle envisageables et les comparer dans le cadre des disjoncteurs haute-tension.

7.3.3 Pré-traitement, génération de maillage, adaptation de maillage

Le solveur a été tellement accéléré que les tâches de préparation géométrique et de réglage de la solution initiale ainsi que les tâches de post-traitement représentent la part majoritaire du temps total nécessaire à la réalisation d'un calcul. Le nouvel outil graphique dont le développement a été initié chez l'industriel par l'auteur de cette thèse, comprend un robuste générateur de maillage de triangles automatique pour la partie 2D et un ensemble d'outils à productivité très importante et optimisée pour la conception des disjoncteurs. Pour MC^3, cette phase était manuelle et manquait de robustesse et de fiabilité pour les complexes géométries de disjoncteurs. Parfois, plusieurs jours étaient nécessaires pour la réalisation d'un maillage qui s'avérait de temps en temps invalide et la prise en main des nouveaux utilisateurs (10 nouveaux par année) requérait environ 3 mois de pratique journalière. La nouvelle conception logicielle, toujours en développement, propose une prise en main en quelques minutes

avec même des profils utilisateurs personnalisables en fonction des projets de développement d'appareils. De plus, les utilisateurs auront une interaction moderne avec la souris : utilisation du clic droit et de la roulette de la souris. L'architecture logicielle sera conçue selon un patron MVC (Model-View-Controler) où le solveur multiphysiques sera exécuté sur un cluster distant. Des extensions seront alors apportées à l'outil conçu dans le cadre de cette thèse pour permettre les communications avec la partie cliente. Concernant la génération des maillages 3D, les outils industriels restent à identifier et représentent un défi technique et scientifique de par la complexité des géométries. Cependant, la technique IBM pourrait éliminer cette problématique et pourrait constituer des travaux futurs. Cela permettrait de réduire les coûts nécessaires à l'adaptation de maillage, voire d'en supprimer le besoin.

7.3.4 Publications scientifiques

La méthode de couplage (chapitre 6) a été accepté par le comité de sélection de la conférence *Gas Dicharge & Applications 2016* en vue d'une présentation orale devant la communauté scientifique spécialiste des disjoncteurs haute-tension. A ce titre un article a été écrit dans le manuscrit correspondant. Un article est en cours d'écriture pour une publication envisagée dans le journal spécialisé en mécanique des fluides numérique *Computer & Fluids*.

RÉFÉRENCES

[1] J. Abanto Florida, "VF++ : un environnement oriente-objet pour la methode des volumes finis." Thèse de doctorat, Ecole Polytechnique de Montreal, 1999.

[2] A. Abdulle, E. Weinan, B. Engquist, et E. Vanden-Eijnden, "The heterogeneous multiscale method", *Acta Numerica*, vol. 21, pp. 1–87, 05 2012.

[3] S. Akhter et J. Roberts, *Multi-core Programming : Increasing Performance Through Software Multi-threading*, série Books by engineers, for engineers. Hillsboro, OR, USA : Intel Press, 2006.

[4] S. Albin, *The Art of Software Architecture : Design Methods and Techniques*, série Wiley Application Development Series. Wiley, 2003.

[5] J. D. Anderson, *Modern Compressible Flow : With Historical Perspective*, 3e éd. McGraw-Hill, 2002.

[6] ANSYS, *Fluent 18.0 user's manual*, 2017.

[7] S. Arabi, J.-Y. Trépanier, et R. Camarero, "A simple extension of roe's scheme for real gases", *Journal of Computational Physics*, vol. 329, pp. 16 – 28, 2017.

[8] L. Bass, P. Clements, et R. Kazman, *Software Architecture in Practice 3rd edition*, série SEI Series in Software Engineering. Pearson Education, 2012.

[9] M. Beaudoin, H. Nilsson, M. Page, R. Magnan, et H. Jasak, "Evaluation of an improved mixing plane interface for openfoam", *IOP Conference Series : Earth and Environmental Science*, vol. 22, no. 2, p. 022004, 2014.

[10] M. S. Benilov, "Understanding and modelling plasma–electrode interaction in high-pressure arc discharges : a review", *Journal of Physics D : Applied Physics*, vol. 41, no. 14, p. 144001, 2008.

[11] A. Bernard-Champmartin, J.-P. Braeunig, et J.-M. Ghidaglia, "An eulerian finite volume solver for multi-material fluid flows with cylindrical symmetry", *Computers and Fluids*, vol. 83, pp. 170 – 176, 2013, numerical methods for highly compressible multi-material flow problems.

[12] R. Bini, N. T. Basse, et M. Seeger, “Arc-induced turbulent mixing in an SF6 circuit breaker model”, *Journal of Physics D : Applied Physics*, vol. 44, no. 2, p. 1301, 2011.

[13] J. Blazek, *Computational Fluid Dynamics : Principles and Applications*, 3e éd. Elsevier, 2015.

[14] J. Bosch, *Design and Use of Software Architectures : Adopting and Evolving a Product-line Approach*, série ACM Press Books. Addison-Wesley, 2000.

[15] E. Braude, *Software design : from programming to architecture.* J. Wiley, 2004.

[16] L. Briggs, W. Miller, et E. Lewis, “Ray-effect mitigation in discrete ordinate-like angular finite element approximations in neutron transport”, *Nuclear Science and Engineering*, vol. 57, p. 205–217, 1975.

[17] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, et M. Stal, *Pattern-Oriented Software Architecture, Volume 1 : A System of Patterns.* Chichester, UK : Wiley, 1996.

[18] J. R. Cary, J. Candy, J. Cobb, R. H. Cohen, T. Epperly, D. J. Estep, S. Krasheninnikov, A. D. Malony, D. C. McCune, L. McInnes, A. Pankin, S. Balay, J. A. Carlsson, M. R. Fahey, R. J. Groebner, A. H. Hakim, S. E. Kruger, M. Miah, A. Pletzer, S. Shasharina, S. Vadlamani, D. Wade-Stein, T. D. Rognlien, A. Morris, S. Shende, G. W. Hammett, K. Indireshkumar, A. Y. Pigarov, et H. Zhang, “Concurrent, parallel, multiphysics coupling in the FACETS project”, *Journal of Physics : Conference Series*, vol. 180, no. 1, p. 012056, 2009.

[19] J. Chai, H. Lee, et S. Patankar, “Treatment of irregular geometries using a cartesian coordinates finite-volume radiation heat transfer procedure”, *Numerical Heat Transfer Part B : Fundamentals*, vol. 26, p. 225–235, 1994.

[20] J. Chen, X. Wang, H. Wang, et J. D. Lee, “Multiscale modeling of dynamic crack propagation”, *Engineering Fracture Mechanics*, vol. 77, no. 4, pp. 736 – 743, 2010.

[21] X. Chen, “Dynamic coupling of a three-dimensional hydrodynamic model with a laterally averaged, two-dimensional hydrodynamic model”, *Journal of Geophysical Research : Oceans*, vol. 112, no. C7, pp. n/a–n/a, 2007, c07022.

[22] T. Christen, “Radiation and nozzle-ablation models for cfd simulations of gas circuit breakers”, dans *Electric Power Equipment - Switching Technology (ICEPE-ST), 2011 1st International Conference on*, Oct 2011, pp. 471–474.

[23] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, et J. Stafford, *Documenting Software Architectures : Views and Beyond*, série SEI Series in Software Engineering. Pearson Education, 2010.

[24] S. S. Dua et C. Ping, "Multi-dimensional radiative transferin non-isothermal cylindrical media with non-isothermal bounding walls", *International Journal of Heat and Mass Transfer*, vol. 18, no. 2, p. 245–259, 1975.

[25] S. Eby, J. Y. Trépanier, et X. Zhang, "Modelling radiative transfer in SF6 circuit-breaker arcs with the P-1 approximation", *Journal of Physics. D, Applied Physics*, vol. 31, no. 13, pp. 1578–1588, 1998.

[26] M. Fowler et K. Beck, *Refactoring : Improving the Design of Existing Code*, série Addison-Wesley object technology series. Addison-Wesley, 1999.

[27] M. Fowler, *Refactoring : Improving the Design of Existing Code*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2002, pp. 256–256.

[28] J. Fořt, J. Fürst, J. Halama, K. Kozel, P. Louda, P. Sváček, Z. Šimka, P. Pánek, et M. Hajsman, "Fvm-fem coupling and its application to turbomachinery", dans *Finite Volumes for Complex Applications VI Problems and Perspectives*, série Springer Proceedings in Mathematics, 2011, vol. 4, pp. 505–512.

[29] T.-P. Fries et H. G. Matthies, "A stabilized and coupled meshfree/meshbased method for the incompressible navier–stokes equations—part i : Stabilization", *Computer Methods in Applied Mechanics and Engineering*, vol. 195, no. 44–47, pp. 6205 – 6224, 2006.

[30] ——, "A stabilized and coupled meshfree/meshbased method for the incompressible navier–stokes equations—part ii : Coupling", *Computer Methods in Applied Mechanics and Engineering*, vol. 195, no. 44–47, pp. 6191 – 6204, 2006.

[31] R. Girard, J. B. Belhaouari, J. J. Gonzalez, et A. Gleizes, "A two-temperature kinetic model of SF6 plasma", *Journal of Physics D : Applied Physics*, vol. 32, no. 22, p. 2890, 1999.

[32] R. Girard, J. J. Gonzalez, et A. Gleizes, "Modelling of a two-temperature SF6 arc plasma during extinction", *Journal of Physics D : Applied Physics*, vol. 32, no. 11, p. 1229, 1999.

[33] P. Glaister, “An approximate linearised riemann solver for the three-dimensional euler equations for real gases using operator splitting”, *Journal of computational physics*, vol. 77, no. 2, pp. 361–383, 1988.

[34] A. Gleizes, T. Robert, J. J. Gonzalez, et A. Pons, “Modelling of the SF6 circuit-breaker arc and of its interaction with the circuit”, *Journal of Physics D : Applied Physics*, vol. 26, no. 9, p. 1439, 1993.

[35] D. Godin et J. Y. Trépanier, “A robust and efficient method for the computation of equilibrium composition in gaseous mixtures”, *Plasma Chemistry and Plasma Processing*, vol. 24, no. 3, pp. 447–473, 2004.

[36] D. Godin, J. Y. Trépanier, M. Reggio, X. Zhang, et R. Camarero, “Modelling and simulation of nozzle ablation in high-voltage circuit-breakers”, *Journal of Physics. D, Applied Physics*, vol. 33, no. 20, pp. 2583–2590, 2000.

[37] J.-J. Gonzalez et P. Freton, “Flow behavior in high-voltage circuit breaker”, *Plasma Science, IEEE Transactions on*, vol. 39, no. 11, pp. 2856–2857, Nov 2011.

[38] T. Gross et D. R. O’Hallaron, *iWarp : Anatomy of a Parallel Computing System*. Cambridge, MA, USA : MIT Press, 1998.

[39] D. N. Gupta, G. N. Patil, D. Srinivas, S. S. Kale, et S. B. Potnis, “Numerical simulation for arc-plasma dynamics during contact opening process in electrical circuit-breakers”, *Journal of Physics : Conference Series*, vol. 208, no. 1, p. 012046, 2010.

[40] D. Guttman, M. Arunachalam, V. Calina, et M. T. Kandemir, “Chapter 21 - prefetch tuning optimizations”, dans *High Performance Parallelism Pearls Volume Two : Multicore and Many-core Programming Approaches*, J. Reinders et J. Jeffers, éds. Boston, MA, USA : Morgan Kaufmann, 2015, vol. 2, pp. 401 – 419.

[41] J. C. Hall, R. Gramunt, et K. Raman, “Chapter 6 - uarch optimization advice”, dans *Intel Xeon Phi Processor High Performance Programming Knights Landing Edition*, J. Reinders, J. Jeffers, et A. Sodani, éds. Boston, MA, USA : Morgan Kaufmann, 2016, pp. 107–145.

[42] C. Hofmeister, R. Nord, et D. Soni, *Applied Software Architecture*, série Addison-Wesley object technology series. Addison-Wesley, 2000.

[43] A. Ilinca, M. Reggio, J. Y. Trépanier, et R. Camarero, “Error estimator and adaptive

moving grids for finite volumes schemes", *AIAA Journal*, vol. 33, no. 11, pp. 2058–2065, 1995.

[44] Intel, *Intel® 64 and IA-32 Architectures Software Developer's Manual volume 1 : Basic Architecture*, 253665e éd., September 2016.

[45] ——, *Intel® 64 and IA-32 Architectures Software Developer's Manual volume 2 : Instruction Set Reference*, 325383e éd., September 2016.

[46] ——, *Intel® Architecture Instruction Set Extensions Programming Reference*, 319433e éd., August 2015.

[47] ——, *Intel® 64 and ia32 Architectures Optimization Reference manual*, 248966e éd., June 2017.

[48] A. A. Iordanidis et C. M. Franck, "Self-consistent radiation-based simulation of electric arcs : Ii. application to gas circuit breakers", *Journal of Physics D : Applied Physics*, vol. 41, no. 13, p. 135206, 2008.

[49] C. Jan, Y. Cressault, A. Gleizes, et K. Bousoltane, "Calculation of radiative properties of SF6 – C2F4 thermal plasmas—application to radiative transfer in high-voltage circuit breakers modelling", *Journal of Physics D : Applied Physics*, vol. 47, no. 1, p. 015204, 2014.

[50] B.-Y. L. K-D Song et K.-Y. Park, "Calculation of the post-arc current in a supersonic nozzle by using the k-e turbulence model", *Journal of the Korean Physical Society*, vol. 45, no. 6, pp. 1537–1543, 2004.

[51] ——, "Calculation of the post-arc current in a supersonic nozzle by using the k-e turbulence model", *Journal of the Korean Physical Society*, vol. 45, no. 6, pp. 1537–1543, 2004.

[52] S. Y. Kadioglu et D. A. Knoll, *An IMEX Method for the Euler Equations that Posses Strong Non-Linear Heat Conduction and Stiff Source Terms (Radiation Hydrodynamics)*, 2011, ch. 13.

[53] D. E. Keyes, L. C. McInnes, C. Woodward, W. Gropp, E. Myra, M. Pernice, J. Bell, J. Brown, A. Clo, J. Connors *et al.*, "Multiphysics simulations challenges and opportunities", *International Journal of High Performance Computing Applications*, vol. 27, no. 1, pp. 4–83, 2013.

[54] Khronos-OpenCL-Working-Group, *The openCL C Specification - Version 2.0*, khronos opencl working group - rev33 éd., April 2016.

[55] D. Kim, J. Larson, et K. Chiu, "Malleable model coupling with prediction", dans *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, May 2012, pp. 360–367.

[56] ——, "Dynamic load balancing for malleable model coupling", dans *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*, July 2012, pp. 150–157.

[57] J. Kim et L. Meadows, "Chapter 18 - exploiting multilevel parallelism in quantum simulations", dans *High Performance Parallelism Pearls Volume Two : Multicore and Many-core Programming Approaches*, J. Reinders et J. Jeffers, éds. Boston, MA, USA : Morgan Kaufmann, 2015, vol. 2, pp. 335 – 354.

[58] M. Y. Kim et S. W. Baek, "Modeling of radiative heat transfer in an axisymmetric cylindrical enclosure with participating medium", *Journal of Quantitative Spectroscopy and Radiative Transfer*, vol. 90, no. 3-4, p. 377–388, 2005.

[59] A. Lani, T. Quintino, D. Kimpe, H. Deconinck, S. Vandewalle, et S. Poedts, "The COOLFluiD framework : design solutions for high performance object oriented scientific computing software", dans *Computational Science–ICCS 2005*, 2005, pp. 279–286.

[60] O. Larrouturou, "How to preserve the mass fractions positivity when computing comprssible multi-component flows", *Research report*, no. RR-1080, 1989, <inria-00075479>.

[61] J. W. Larson, "Some organising principles for coupling in multiphysics and multiscale models", *Preprint ANL/MCS-P1414-0207, Mathematics and Computer Science Division, Argonne National Laboratory*, 2006.

[62] J. Larson, R. Jacob, et E. Ong, "The model coupling toolkit : A new fortran90 toolkit for building multiphysics parallel coupled models", *International Journal of High Performance Computing Applications*, vol. 19, no. 3, pp. 277–292, 2005.

[63] J. W. Larson, "Ten organising principles for coupling in multiphysics and multiscale models", *ANZIAM Journal*, vol. 48, pp. 1090–1111, 2009.

[64] J.-F. Lemieux, D. A. Knoll, M. Losch, et C. Girard, "A second-order accurate in time

implicit–explicit {IMEX} integration scheme for sea ice dynamics", *Journal of Computational Physics*, vol. 263, no. 0, pp. 375 – 392, 2014.

[65] M.-S. Liou et C. J. Steffen Jr, "A new flux splitting scheme", *Journal of Computational physics*, vol. 107, no. 1, pp. 23–39, 1993.

[66] J.-Y. Liu, M. R. Smith, F.-A. Kuo, et J.-S. Wu, "Hybrid OpenMP/AVX acceleration of a Split HLL finite volume method for the shallow water and euler equations", *Computers and Fluids*, vol. 110, pp. 181 – 188, 2015, parCFD 2013.

[67] C. Lüders, T. Suwanasri, et R. Dommerque, "Investigation of an SF6 -selfblast circuit breaker", *Journal of Physics D : Applied Physics*, vol. 39, no. 4, p. 666, 2006.

[68] G. Marquezin, L. Yi, Z. Deng, P. Robin-Jouan, et J. Rodriguez, "Contribution of the numerical simulation tools in the high voltage circuit-breaker conception", dans *Electric Power Equipment - Switching Technology (ICEPE-ST), 2011 1st International Conference on*, Oct 2011, pp. 370–374.

[69] A. Martin, J. Y. Trépanier, M. Reggio, et X. Guo, "Transient ablation regime in circuit breakers", *Plasma Science and Technology*, vol. 9, no. 6, pp. 653–656, 2007.

[70] A. Martin, M. Reggio, et J. Y. Trépanier, "Numerical solution of axisymmetric multi-species compressible gas flow : Towards improved circuit breaker simulation", *International Journal of Computational Fluid Dynamics*, vol. 22, no. 4, pp. 259–271, 2008.

[71] P. Maruzewski, A. Martin, M. Reggio, et J. Y. Trépanier, "Simulation of arc-electrode interaction using sheath modelling in SF6 circuit-breakers", *Journal of Physics. D, Applied Physics*, vol. 35, no. 9, pp. 891–899, 2002.

[72] M. McCool, A. Robison, et J. Reinders, *Structured Parallel Programming : Patterns for Efficient Computation*. Boston, MA, USA : Morgan Kaufmann Publishers Inc., 2012.

[73] M. McCool, A. D. Robison, et J. Reinders, "Chapter 3 - patterns", dans *Structured Parallel Programming*, M. McCool, A. D. Robison, et J. Reinders, éds. Boston : Morgan Kaufmann, 2012, pp. 79 – 119.

[74] ——, "Chapter 6 - data reorganization", dans *Structured Parallel Programming*, M. McCool, A. D. Robison, et J. Reinders, éds. Boston : Morgan Kaufmann, 2012, pp. 179 – 197.

[75] ——, “Chapter 8 - fork–join”, dans *Structured Parallel Programming*, M. McCool, A. D. Robison, et J. Reinders, éds. Boston : Morgan Kaufmann, 2012, pp. 209 – 251.

[76] M. Melot, “Modelisation numerique du transfert radiatif par la methode des volumes finis dans les disjoncteurs a SF6”, *Mémoire de Maitrise - Sciences appliquées*, 2009.

[77] S. Miller, R. Campbell, C. Elsworth, J. Pitt, et D. Boger, “An overset grid method for fluid-structure interaction”, *World Journal of Mechanics*, vol. 4, pp. 217–237, 2014.

[78] S. K. Mishra, “Development of a multiscale and multiphysics simulation framework for reaction-diffusion-convection problems”, p. 342, 2009.

[79] M. Modest, *Radiative Heat Transfer*, 2e éd. Academic Press, 2003.

[80] H. Nordborg et A. A. Iordanidis, “Self-consistent radiation based modelling of electric arcs : I. efficient radiation approximations”, *Journal of Physics D : Applied Physics*, vol. 41, no. 13, p. 135205, 2008.

[81] Nvidia, *CUDA C best practices guide*, dg-05603-001 v8.0 éd., Nvidia, January 2017.

[82] ——, *CUDA C programming guide*, pg-02829-001 v8.0 éd., Nvidia, January 2017.

[83] E. O’Brien, “Chapter 17 - coarse-grained openmp for scalable hybrid parallelism”, dans *High Performance Parallelism Pearls Volume Two : Multicore and Many-core Programming Approaches*, J. Reinders et J. Jeffers, éds. Boston, MA, USA : Morgan Kaufmann, 2015, vol. 2, pp. 321 – 334.

[84] OMG, *OMG Unified Language Modeling (OMG UML) - Version 2.5*, Mars 2015.

[85] K. Oßwald, A. Siegmund, P. Birken, V. Hannemann, et A. Meister, “L2roe : a low dissipation version of Roe’s approximate riemann solver for low mach numbers”, *International Journal for Numerical Methods in Fluids*, vol. 81, no. 2, pp. 71–86, 2016, fld.4175.

[86] M. Paraschivoiu, “An adaptive method for the exact resolution of shock waves and contact discontinuities”, 07 1993.

[87] J. H. Park, K. H. Kim, C. H. Yeo, et H. K. Kim, “CFD analysis of arc-flow interaction in a high-voltage gas circuit breaker using an overset method”, *Plasma Science, IEEE Transactions on*, vol. 42, no. 1, pp. 175–184, Jan 2014.

[88] J. Patel, H. Park, C. de Oliveira, et D. Knoll, “Efficient multiphysics coupling for fast burst reactors in slab geometry”, *Journal of Computational and Theoretical Transport*, no. ahead-of-print, pp. 1–25, 2014.

[89] A. Pedersen, “On the electrical breakdown of gaseous dielectrics”, *IEEE Transactions on Electrical Insulation*, vol. 24, no. 5, pp. 721–739, 1989.

[90] M. Pelanti, “Low Mach number preconditioning techniques for roe-type and HLLC-type methods for a two-phase compressible flow model”, *Applied Mathematics and Computation*, vol. 310, pp. 112 – 133, 2017.

[91] H. Pellegrin, , J. Y. Trépanier, R. Camarero, et X. Zhang, “Computation of the self-induced magnetic field in circuit-breaker arcs”, *IEEE Transactions on Plasma Science*, vol. 25, no. 5, pp. 974–981, 1997.

[92] E. Petro et J.-Y. Trepanier, “MC3 v5.3 : Validation de la méthodologie hybride”, Ecole Polytechnique de Montreal, Rapp. tech., November 2010.

[93] T. M. Pigoski, *Practical Software Maintenance : Best Practices for Managing Your Software Investment*, 1er éd. Wiley Publishing, 1996.

[94] A. PRAMANIK, *ELECTROMAGNETISM : Theory and Applications*, 2e éd. PHI Learning Pvt. Ltd., 2008.

[95] ——, *ELECTROMAGNETISM : Problems with Solutions*, 3e éd. PHI Learning Pvt. Ltd., 2012.

[96] G. Raithby et E. Chui, “A finite-volume method for predicting a radiant heat transfer in enclosures with participating media”, *Journal of Heat Transfer : Transactions of the ASME*, vol. 112, p. 415–423, 1990.

[97] H. Randrianandraina, Y. Cressault, et A. Gleizes, “Improvements of radiative transfer calculation for SF6 thermal plasmas”, *Journal of Physics D : Applied Physics*, vol. 44, no. 19, p. 194012, 2011.

[98] G. M. Raskulinec et E. Fiksman, “Chapter 22 - simd functions via openmp”, dans *High Performance Parallelism Pearls Volume Two : Multicore and Many-core Programming Approaches*, J. Reinders et J. Jeffers, éds. Boston, MA, USA : Morgan Kaufmann, 2015, vol. 2, pp. 421 – 440.

[99] J. Reinders, “Chapter 8 - tasks and threads”, dans *Intel Xeon Phi Processor High Performance Programming Knights Landing Edition*, J. Reinders, J. Jeffers, et A. Sodani, éds. Boston, MA, USA : Morgan Kaufmann, 2016, pp. 155–172.

[100] ——, “Chapter 9 - vectorization”, dans *Intel Xeon Phi Processor High Performance Programming Knights Landing Edition*, J. Reinders, J. Jeffers, et A. Sodani, éds. Boston, MA, USA : Morgan Kaufmann, 2016, pp. 173–212.

[101] ——, *Intel Threading Building Blocks : Outfitting C++ for Multi-core Processor Parallelism*. Sebastopol, CA, USA : Intel Press, 2007.

[102] J. Reinders et J. Jeffers, éds., *High Performance Parallelism Pearls : Multicore and Many-core Programming Approaches*. Boston, MA, USA : Morgan Kaufmann Publishers Inc., 2015, vol. 2.

[103] J. Reinders, A. Jha, et S. Li, “Chapter 12 - vectorization with avx-512 intrinsics”, dans *Intel Xeon Phi Processor High Performance Programming Knights Landing Edition*, J. Reinders, J. Jeffers, et A. Sodani, éds. Boston, MA, USA : Morgan Kaufmann, 2016, pp. 269–296.

[104] F. Rieper, “A low-mach number fix for Roe’s approximate riemann solver”, *Journal of Computational Physics*, vol. 230, no. 13, pp. 5263 – 5287, 2011.

[105] P. Robin-Jouan, “Peut-on faire confiance à la simulation numérique de la coupure haute-tension pour développer les disjoncteurs du futur ?” *Revue de l’Électricité et de l’Électronique*, vol. 11, pp. 36–40, 2004.

[106] P. L. Roe, “Approximate riemann solvers, parameter vectors, and difference schemes”, *Journal of Computational Physics*, vol. 43, pp. 357–372, 1981.

[107] D. W. I. Rouson, H. Adalsteinsson, et J. Xia, “Design patterns for multiphysics modeling in fortran 2003 and c++”, *ACM Trans. Math. Softw.*, vol. 37, no. 1, pp. 3 :1–3 :30, Jan. 2010.

[108] ——, “Design patterns for multiphysics modeling in FORTRAN 2003 and C++”, *ACM Trans. Math. Softw.*, vol. 37, no. 1, pp. 3 :1–3 :30, Jan. 2010.

[109] C. B. Ruchti et L. Niemeyer, *IEEE Transactions on Plasma Science*, no. 14, pp. 423–434.

[110] J. Ryan, L. Halpern, et M. Borrel, “Domain decomposition vs. overset chimera grid approaches for coupling CFD and CAA”, *Seventh International Conference on Computational Fluid Dynamics (ICCFD7)*, 2012.

[111] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Society for Industrial Mathematics, 2003.

[112] M. Sardella, “On a coupled finite element-finite volume method for convection-diffusion problems”, *IMA Journal of Numerical Analysis*, vol. 20, no. 2, pp. 281–301, 2000.

[113] T. Schwarz, F. Spiering, et N. Kroll, “Grid coupling by means of chimera interpolation techniques”, dans *Second Symposium ”Simulation of Wing and Nacelle Stall”*, 2010.

[114] M. Seeger, B. Galletti, R. Bini, V. Dousset, A. Iordanidis, D. Over, N. Mahdizadeh, M. Schwinne, P. Stoller, et T. Votteler, “Some aspects of current interruption physics in high voltage circuit breakers”, *Contributions to Plasma Physics*, vol. 54, no. 2, pp. 225–234, 2014.

[115] P. Solin, J. Cerveny, L. Dubcova, et D. Andrs, “Monolithic discretization of linear thermoelasticity problems via adaptive multimesh -fem”, *Journal of Computational and Applied Mathematics*, vol. 234, no. 7, pp. 2350 – 2357, 2010, fourth International Conference on Advanced {COmputational} Methods in {ENgineering} (ACOMEN 2008).

[116] P. Stoller, M. Seeger, A. Iordanidis, et G. Naidis, “CO2 as an arc interruption medium in gas circuit breakers”, *Plasma Science, IEEE Transactions on*, vol. 41, no. 8, pp. 2359–2369, 2013.

[117] Y. Tanaka et K. Suzuki, “Development of a chemically nonequilibrium model on decaying SF6 arc plasmas”, *Power Delivery, IEEE Transactions on*, vol. 28, no. 4, pp. 2623–2629, 2013.

[118] T. J. Tautges et A. Caceres, “Scalable parallel solution coupling for multiphysics reactor simulation”, *Journal of Physics : Conference Series*, vol. 180, no. 1, p. 012017, 2009.

[119] B. Thornber, A. Mosedale, D. Drikakis, D. Youngs, et R. Williams, “An improved reconstruction method for compressible flows with low mach number features”, *Journal of Computational Physics*, vol. 227, no. 10, pp. 4873 – 4894, 2008.

[120] E. F. Toro, *Riemann Solvers and Numerical Methods for Fluid Dynamics, A Practical Introduction*. Springer, 2009.

[121] J.-Y. Trepanier, M. Reggio, Y. Lauze, et R. Jeanjean, “Analysis of the dielectric strength of an SF6 circuit breaker”, *Power Delivery, IEEE Transactions on*, vol. 6, no. 2, pp. 809–815, 1991.

[122] J. Y. Trépanier, M. Reggio, H. Zhang, et R. Camarero, “A finite-volume method for the euler equations on arbitrary lagrangian-eulerian grids”, *Comput. Fluids*, vol. 20, no. 4, pp. 399–409, 1991.

[123] J. Y. Trépanier, M. Reggio, M. Paraschivoiu, et R. Camarero, “Unsteady euler solutions for arbitrarily moving bodies and boundaries”, *AIAA Journal*, vol. 31, no. 10, pp. 1869–1876, 1993.

[124] B. Vermeire et S. Nadarajah, “Adaptive {IMEX} schemes for high-order unstructured methods”, *Journal of Computational Physics*, vol. 280, no. 0, pp. 261 – 286, 2015.

[125] H. K. Versteeg et W. Malalasekera, *An Introduction to Computational Fluid Dynamics : The Finite Volume Method*, 2e éd. Pearson Education Limited, 2007.

[126] Z. J. Wang et V. Parthasarathy, “A fully automated chimera methodology for multiple moving body problems”, *International Journal for Numerical Methods in Fluids*, vol. 33, no. 7, pp. 919–938, 2000.

[127] H. Weller, S.-J. Lock, et N. Wood, “Runge–kutta {IMEX} schemes for the horizontally explicit/vertically implicit (hevi) solution of wave equations”, *Journal of Computational Physics*, vol. 252, no. 0, pp. 365 – 381, 2013.

[128] Y. Wu, X. Xie, et L. Chen, “Hybrid stress finite volume method for linear elasticity problems”, *INTERNATIONAL JOURNAL OF NUMERICAL ANALYSIS AND MODELING*, vol. 10, no. 3, pp. 634–656, 2013.

[129] J. D. Yan, K. I. Nuttall, et M. T. C. Fang, “A comparative study of turbulence models for SF6 arcs in a supersonic nozzle”, *Journal of Physics D : Applied Physics*, vol. 32, no. 6, p. 1401, 1999.

[130] A.-J. N. Yzelman, D. Roose, et K. Meerbergen, “Chapter 27 - sparse matrix-vector multiplication : Parallelization and vectorization”, dans *High Performance Parallelism Pearls : Multicore and Many-core Programming Approaches*, J. Reinders et J. Jeffers, éds. Boston, MA, USA : Morgan Kaufmann, 2015, vol. 1, pp. 457 – 476.

[131] H. Zhang, M. Reggio, J. Y. Trépanier, et R. Camarero, “Discrete form of the GCL

for moving meshes and its implementation in CFD schemes", *Comput. Fluids*, vol. 22, no. 1, pp. 9–23, 1993.

[132] X. Zhang, , J. Y. Trépanier, et R. Camarero, "Modelling and computation of arc-flow interaction in circuit-breaker", *International Journal of Computational Fluid Dynamics*, vol. 2, pp. 41–64, 1994.

[133] ——, "Numerical simulation of a 2 kA convection-stabilized nitrogen arc using CFD tools", *Journal of Physics. D,Applied Physics*, vol. 30, no. 23, pp. 3240–3252, 1997.

[134] Z. Zou, J. Liu, W. Zhang, et P. Wang, "Shroud leakage flow models and a multidimensional coupling cfd (computational fluid dynamics) method for shrouded turbines", *Energy*, vol. 103, pp. 410 – 429, 2016.

[135] S. P. Zwart, S. McMillan, S. Harfst, D. Groen, M. Fujii, B. Ó. Nualláin, E. Glebbeek, D. Heggie, J. Lombardi, P. Hut *et al.*, "A multiphysics and multiscale software environment for modeling astrophysical systems", *New Astronomy*, vol. 14, no. 4, pp. 369 – 378, 2009.

ANNEXE A LE RAYONNEMENT THERMIQUE DANS LES GAZ ET DANS LES PLASMAS

Nature du rayonnement thermique dans les gaz et les plasmas :
Le rayonnement thermique est un transfert d'énergie qui contrairement à la conduction et à la convection de chaleur, ne requièrent pas la présence d'un milieu de propagation. Ainsi, il peut dans ce cas, se transmettre à une très grande distance, le Soleil en est le parfait exemple. Le rayonnement thermique est souvent assimilé à un transfert par ondes électromagnétiques, dont les fréquences et l'intensité de l'émission dépendent de la température du corps émetteur. Lorsqu'un corps atteint la température du zéro absolu alors il n'émet plus aucune onde électromagnétique ou photons. La dépendance du phénomène vis-à-vis de la température le distingue également de la conduction et de la convection. En effet, dans le cas du rayonnement thermique, le flux de chaleur transmis n'est plus linéairement proportionnel à une différence de température mais proportionnel à une différence de température à l'ordre quatre ($q \propto T^4 - T_\infty^4$). Cela montre bien que le rayonnement devient le transfert d'énergie dominant au fur et à mesure que la température croît. Par conséquent, la modélisation du transfert radiatif dans des applications telles que la modélisation des réacteurs nucléaires, des combustions, des plasmas, et de bien d'autres, joue un rôle primordial.

Suivant l'application, dans un milieu solide, liquide ou gazeux, on peut assimiler le rayonnement thermique comme un ensemble d'ondes électromagnétiques ou bien comme un ensemble de photons (particules sans masse). Bien que les deux méthodes soient interchangeables, ce dernier cas se prête mieux aux prédictions des propriétés radiatives dans les gaz. Les vitesses de propagation c, dépendent de l'indice de réfraction du milieu η et de la vitesse de la lumière dans le vide c_0 et se calculent telles que : $c = \frac{c_0}{\eta}$. Dans le cas de la propagation dans le vide, la vitesse est égale à la vitesse de la lumière ($\eta = 1$), tandis que dans les gaz la vitesse est très légèrement plus faible (η est très proche et supérieur à 1). Schématiquement, on peut voir le transfert par rayonnement comme un ensemble de rayons qui se propagent dans un milieu (ou sans milieu). A la rencontre d'un autre milieu ou d'une autre particule, ils peuvent être totalement réfléchis ou seulement l'être partiellement et dans ce cas, une partie pénètre le milieu. En se propageant dans ce milieu, l'onde peut être atténuée par absorption. Si l'absorption est complète, le milieu est dit *opaque* tandis que si elle est nulle, le milieu est dit *transparent*. En pratique, on rencontre beaucoup de milieux qui sont *semi-transparents*, sachant que le niveau de transparence dépend aussi bien du matériau lui-même que des longueurs d'ondes qui composent le rayonnement et de l'épaisseur considérée. Cependant, certains milieux tels

que la plupart des métaux sont totalement opaques (l'épaisseur du matériau est très déterminante dans le cas des métaux. Par exemple, les visières des casques des astronautes (Figure A.1) sont recouvertes d'une très fine couche d'or, de l'ordre de quelques microns seulement. Cela permet ainsi de filtrer les rayonnements UV, pour lesquels elles sont opaques, tout en laissant passer une partie du spectre visible pour lequel elles sont transparentes. Les casques des pompiers (Figure A.1) sont également équipés de filtres du même type pour limiter la transmission de chaleur par rayonnement au niveau du visage).

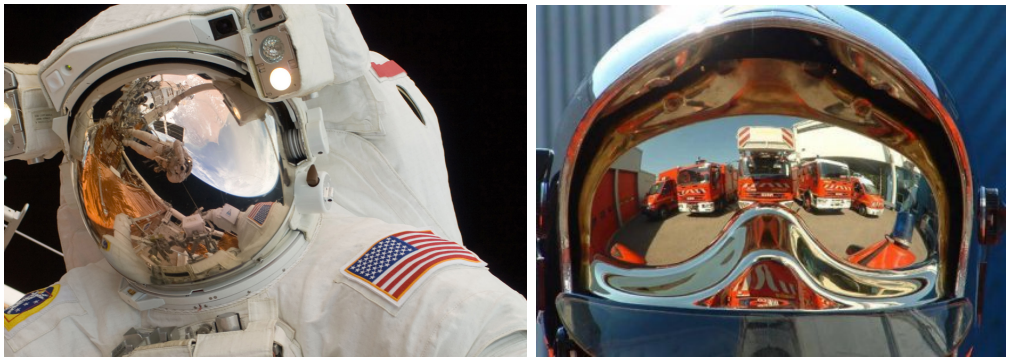


Figure A.1 Exemple d'application de filtres à rayonnement - casque d'astronaute, casque de pompier

Au même titre que tout corps solide, les milieux gazeux peuvent également absorber et émettre du rayonnement, ils sont alors dits : *participants*. D'après la mécanique quantique, chaque molécule ou atome peut voir son niveau d'énergie varier, en recevant de l'énergie sous forme de photons (absorption) ou en libérer en émettant un photon. Cependant, seul un nombre discret de niveaux d'énergie sont possibles, ce qui implique des émissions et des absorptions à des fréquences bien précises. De plus, les gaz émettent selon certaines bandes de fréquences qui leur sont caractéristiques : le spectre est dit *sélectif*. Comme le milieu est participant, l'intensité radiative peut être atténuée par absorption ou par diffusion ("out-scattering"). La capacité d'absorption que possède un gaz selon une épaisseur et une longueur d'onde données s'appelle l'épaisseur optique τ. Si $\tau >> 1$ le milieu est dit optiquement épais, à l'inverse si $\tau << 1$, il est considéré optiquement fin. La perte par diffusion est transférée vers une autre direction (gain pour cette dernière). Alors qu'il traverse le milieu participant, un rayon reçoit aussi de l'énergie provenant d'autres directions (par émission et absorption "in-scattering").

Dans le cas des plasmas, le comportement est encore plus complexe. En effet, l'état fortement ionisé du gaz induit des phénomènes de collisions au sein du plasma. Dans l'hypothèse de l'équilibre thermodynamique local, ces collisions sont la cause principale du rayonnement.

Elles sont principalement la combinaison de trois phénomènes : *le rayonnement d'excitation*, *la recombinaison et l'attachement radiatif*, et *le rayonnement de freinage*.

Rayonnement d'excitation :

Le rayonnement d'excitation, appelé aussi, la transition lié-lié, concerne les électrons. Lorsqu'ils changent spontanément de niveau d'énergie, la transition provoque l'émission ou l'absorption d'un photon. Passant d'un état d'excitation à un autre, seulement à certaines longueurs d'ondes bien précises, le rayonnement se caractérise sous forme de raies, c'est à dire d'un spectre discontinu. Cela rend l'intégration sur tout le domaine spectral difficile.

Recombinaison et attachement radiatif :

La recombinaison radiative et l'attachement radiatif, appelés aussi, la transition lié-libre, concerne quant à elle l'interaction entre les électrons, les ions et les atomes neutres. L'état du plasma est aussi le fruit de recombinaisons et de dissociations atomiques, qui lorsqu'elles interviennent produisent également du rayonnement. Un ion peut capturer un électron libre circulant dans son champ, produisant une recombinaison, c'est à dire également un dégagement d'énergie sous forme de rayonnement, lié à l'énergie cinétique de l'électron libre et à son énergie de liaison. L'attachement radiatif émet également du rayonnement. Il se produit lors de la formation d'un ion négatif, quand un électron vient se combiner avec un atome neutre. Que ce soit dans le cas d'une recombinaison radiative ou dans le cas d'un attachement radiatif, un spectre continu de rayonnement en résulte.

Rayonnement de freinage :

Lors de la collision d'un électron avec un ion ou un atome, la combinaison n'est pas systématique. En effet, un électron libre peut seulement changer de trajectoire et de vitesse, influencé par le champ électrique d'un ion ou d'un atome. Une partie de l'énergie cinétique de l'électron libre est alors perdue sous forme de rayonnement (émission d'un photon). On parle alors de rayonnement dit : de freinage (les électrons sont freinés). Dans ce cas, le spectre de rayonnement est continu.

Grandeurs caractéristiques et équations du transfert radiatif (RTE)

Intensité radiative :

Le concept d'intensité radiative permet de caractériser la quantité d'énergie radiative dans une direction donnée. L'intensité radiative spectrale, notée I_λ, est la quantité d'énergie par unité de temps émise par unité de surface projetée perpendiculairement à la direction considérée et contenue dans un angle solide unitaire. Elle s'exprime en $[W.m^{-2}.sr^{-1}.\mu m^{-1}]$. Elle dépend donc du point d'observation, de la direction et de la longueur d'onde λ et lorsque qu'intégrée sur tout le spectre on parle d'intensité radiative totale (en $[W.m^{-2}.sr^{-1}]$).

Rayonnement incident :

Lorsque intégrée dans toutes les directions, l'intensité radiative (spectrale ou totale) permet de calculer le rayonnement incident (respectivement spectral ou total), noté G . L'unité du rayonnement incident total s'exprime en $[W.m^{-2}]$.

Flux radiatif :

Le flux de chaleur radiatif traversant un élément de surface de normale $\hat{n}$ dans une direction donnée $\hat{s}$ (et intégré sur tout l'espace de l'angle solide) est définit par :

$$q \cdot \hat{n} = \int_{4\pi} I(r, \hat{s})\hat{s} \cdot \hat{n} \; d\Omega \tag{A.1}$$

où Ω représente l'angle solide, qui se simplifie en :

$$q = \int_{4\pi} I(r, \hat{s})\hat{s} \; d\Omega \tag{A.2}$$

Cette relation constitue le flux de chaleur radiatif (qui est un vecteur) et s'exprime en $[W.m^{-2}]$. Le terme source d'arc dû au rayonnement ($\nabla .\; q$) contribue à l'équation de conservation de l'énergie (3.1).

Équation de transfert radiatif (RTE) :

L'équation de transfert radiatif exprime la conservation de l'énergie radiative le long d'un rayon lumineux dans la direction $\hat{s}$. Dans un milieu participatif de l'énergie est obtenue par émission et diffusion "in-scattering", tandis qu'un peu d'énergie est perdue par absorption et diffusion "out-scattering". Selon la définition donnée par Modest [79], l'équation de conservation, en négligeant la dépendance temporelle, s'écrit :

$$\left.\frac{dI_\nu}{ds}\right|_{\hat{s}} = \nabla I_\nu|_{\hat{s}} \cdot \hat{s} = \kappa_\nu I_{b\nu} - \beta_\nu I_\nu + \frac{\sigma_{s\nu}}{4\pi} \int_{4\pi} I_\nu(\hat{s}_i)\Phi(\hat{s}_i, \hat{s}) d\Omega_i \tag{A.3}$$

où I_ν est l'intensité radiative spectrale dans une direction $\hat{s}$ à la fréquence ν. κ_ν est le coefficient d'absorption et $\sigma_{s\nu}$ est le coefficient de diffusion. Ce qui permet de définir le coefficient d'extinction $\beta_\nu = \kappa_\nu + \sigma_{s\nu}$. Enfin, $\Phi(\hat{s}_i, \hat{s})$ est la fonction de phase. La difficulté pour résoudre l'équation (A.3) vient du fait que l'intensité radiative I_ν dépend de six variables : trois dans l'espace, deux pour la direction $\hat{s}$ et une pour la fréquence ν.

Dans le contexte des disjoncteurs, et des plasmas thermique de manière générale, la diffusion est négligée, ce qui permet de simplifier l'équation A.3.

ANNEXE B ANALYSE GLOBALE DÉTAILLÉE AVEC MÉTRIQUES CPU

Dans cette section, une analyse détaillée des métriques processeurs est présentée. Afin de conserver le sens et la compréhension des notions techniques, les termes employés sont conservés sous leurs appellations anglophones. Le solveur MC^3 est composé d'environ 260 subroutines fortran. Pour chaque métrique, on présentera les résultats seulement pour les fonctions dont l'impact sur la vitesse d'exécution globale est suffisamment important. Cela est réalisable grâce à une connaissance globale avancée du code de calcul, pour optimiser la vitesse d'exécution d'un programme il faut en avoir une excellente vision d'ensemble, d'autant plus pour des codes de calculs multi-physiques. En effet, certaines fonctions peuvent être qualifiées de très mauvaises selon certaines métriques mais représentent un nombre de cycles très petits. Comme on cherche à accélérer le code en minimisant l'effort de modification, il faut d'abord s'intéresser aux fonctions les plus consommatrices de ressources puisque cela permettra d'obtenir les plus gros gains potentiels de vitesse. De plus, les analyses de chaque métrique ne sont pas indépendantes les unes des autres même si présentée comme telles dans les prochains paragraphes.

Cette section s'appuie sur le manuel d'optimisation des architectures 64 bits Intel [47] pour la référence documentaire et sur l'expertise de l'auteur de cette thèse.

A. Mesures de performance du Front-End

Cette section reflète plutôt des mesures concernant le débit d'instructions, leurs types et de leurs impacts sur l'exécution du code.

ICache Misses :

Catégorie : Front-End Bound
Sous-catégorie : Front-End Latency

Cette métrique révèle les parties du code pour lesquels le CPU a attendu la réception des instructions qui n'étaient pas disponibles dans le cache de niveau 1. La vitesse d'exécution est considérablement ralentie, il y a trop d'instructions à traiter, ce qui sature l'espace disponible dans le cache d'instructions de niveau 1. La table B.1 indique pour les fonctions problématiques le pourcentage de cycles sur le nombre total de cycle durant lequel le processeur attendait (sans rien exécuter).

Tableau B.1 Résumé des mesures d'*ICache Misses*

Fonctions	Catégories	Valeurs	Impacts
cfpdts	fluide	9.7 %	critique
cfpgas	fluide	5.7 %	critique
intptp	rad	5.6 %	critique
cfstmc	maillage	4.7 %	critique
cfpror	fluide	0.333 %	critique
log	fluide	0.3 %	critique

Dans le cas de ces fonctions, soit le chemin d'exécution est trop incertain pour que les unités de prédiction puissent le prédire correctement et ainsi alimenter les bonnes instructions aux bonnes unités d'exécution au moment opportun ; soit le cache est trop pollué par des instructions inutiles mais encore maintenues dans le cache ; soit trop d'instructions interdépendantes sont à réalisées simultanément. Cela traduit généralement une mauvaise conception algorithmique, en d'autres mots, le code est trop compliqué et/ou trop fragmenté pour alimenter efficacement le processeur. Concernant les fonctions entrant en jeu, une réécriture doit être faite pour simplifier le débit d'instructions, les portions de code inutiles doivent être éliminées, les algorithmes naïfs doivent être remplacés par des algorithmes plus efficaces. C'est donc un problème de conception et d'écriture du code source. Ce type d'événement doit absolument être évité dans un code de calcul à haute performance, l'impact est donc qualifié de critique pour toutes les fonctions, les outils de profilage permettent d'identifier les fonctions en cause.

ITLBs Overhead :

Catégorie : Front-End Bound

Sous-catégorie : Front-End Latency

Tableau B.2 Résumé des mesures d'*ITLBs Overhead*

Fonctions	Catégories	Valeurs	Impacts
cfswno	maillage	14.4 %	critique
cfparc	modèle arc	4.1 %	critique
intptp	rad	1.7 %	critique
intvol	rad	0.1 %	critique

Les **I**nstruction **T**ranslation **L**ook-aside **B**uffers, ou ITLBs, sont des tampons qui permettent d'accélérer la traduction d'adresse virtuelle en adresse physique. Lorsque le processeur demande une adresse (virtuelle), elle doit être traduite en adresse physique pour réellement pouvoir accéder à la donnée qui y est présente. Afin de ne pas surcharger l'unité de génération d'adresse (**AGU**) qui traduit les adresses, les ITLBs stockent les adresses récemment traduites selon des pages (équivalent à une sous-hiérarchie). Cependant, si l'adresse suivante demandée est loin (en mémoire), l'impact est très mauvais sur les performances car elle n'est potentiellement pas présente dans la même page ou dans les ITLBs et il faut alors relire la table directement en DRAM principale. La table B.2 présente une estimation de la pénalité due aux changements de page.

Dans notre cas, trop d'instructions sont lues aléatoirement et à des instants très rapprochés c'est à dire à des adresses aléatoires et donc très différentes les unes des autres ce qui force des changements de pages (stockées en mémoire DRAM principale). C'est un événement à fortement éviter dans un code de calcul à haute performance puisque cela peut rendre indisponibles les instructions au moment où elles seront requises et peut causer des ralentissements pour les autres parties du programme. Le problème peut être résolu en révisant les algorithmes et la structure du code source. Les instructions (fonctions) utilisées ensemble doivent être conservées proches en mémoire et elles sont actuellement trop parsemées entre les modules.

Branch Resteers :

Catégorie : Front-End Bound

Sous-catégorie : Front-End Latency

L'**U**nité de **P**rédiction de **B**ranches (**BPU**) du cœur cherche à prédire à l'avance quelle(s) branche(s) seront prises pour ainsi essayer de commencer à charger et à exécuter des instructions à l'avance. Cependant, lorsque le chemin est trop compliqué et la prédiction trop difficile, le processeur assume la(les) première(s) branche(s) comme prise et charge les instructions et les données correspondantes. Lorsqu'il s'est trompé, les instructions et les données chargées ne sont pas les bonnes, le pipeline est alors complètement vidé et toutes les instructions et les données doivent être rechargées en direct. Cette métrique représente le nombre de cycles dans la fonction associée durant lesquels le processeur attendait la réalimentation du pipeline sans rien exécuter.

Tableau B.3 Résumé des mesures de *Branch Resteers*

Fonctions	Catégories	Valeurs	Impacts
intptp	rad	100.0 %	critique
intsurf2	rad	95.8 %	critique
adusnb	maillage	67.6 %	fort
gpskcd	renum.	20.5 %	fort
arceq2	rad	16.2 %	critique
gpskcm	renum.	12 %	fort
ddot	AX=Y	11.6 %	fort
gpskcc	renum.	9.7 %	fort
renumber	renum.	9.3 %	fort
crs2sky	AX=Y	9.3 %	critique
adxlap	maillage	8.7 %	critique
cfptsi	fluide	8.3 %	fort
gpskck	renum.	7.9 %	fort
arcprt	modèle arc	7.9 %	fort
aduptp	maillage	7.5 %	fort
sluss	AX=Y	7.2 %	fort
arcsmt	modèle arc	7.1 %	fort
cfsocs	maillage	5.9 %	fort
advord	maillage	5.4 %	médium

Trop de branches, des algorithmes trop compliqués ou naïfs et un style de code source inapproprié sont les causes des ralentissements importants dans ces fonctions qui doivent être totalement éliminées ou réécrites avec des changements algorithmiques majeurs pour minimiser le nombres de branches et le niveau d'imbrication de certaines branches.

DSB Switches :

Catégorie : Front-End Bound

Sous-catégorie : Front-End Latency

Le **DSB**, ou le **D**ecoded **S**tream **B**uffer, est un tampon d'instructions déjà décodées. Il évite l'utilisation du pipeline de décodage (MITE) lorsque des instructions ont déjà été décodées. Cependant, lorsque ce n'est pas le cas, le processeur change du mode DSB au mode MITE introduisant une pénalité estimé par cette métrique. Dans notre cas, les tampons sont trop petits, il y a trop d'instructions.

Tableau B.4 Résumé des mesures de *DSB Switches*

Fonctions	Catégories	Valeurs	Impacts
advind	maillage	52.3 %	medium
gasrad	rad	33.8 %	medium
sort	renum.	21.3 %	medium
intptp	rad	12.8 %	medium
pow	fluide	8.5 %	fort
matcrs	AX=Y	4.8 %	fort
cfprgl	fluide	4.0 %	fort
log	fluide	3.0 %	fort
cfpror	fluide	1.0 %	fort

MS Switches :

Catégorie : Front-End Bound

Sous-catégorie : Front-End Latency

Le troisième pipeline d'instructions est le **M**icrocode **S**equencer (**MS**). Le processeur n'est pas optimisé pour obtenir les instructions depuis ce pipeline et il y a donc un coût associé au saut vers ce pipeline ainsi qu'au temps de traitement de ce pipeline pour produire les micro-opérations traitées par les autres unités d'exécution. Certaines opérations ne peuvent pas être réalisées par les deux autres pipelines (**MITE** et **DSB**), comme par exemples le traitement des exceptions en calcul avec nombres flottants (division par zéro, dépassement de valeurs...).

Tableau B.5 Résumé des mesures de *MS Switches*

Fonctions	Catégories	Valeurs	Impacts
srcrad	rad	40.0 %	critique
fction1	rad	8.5 %	critique
ptgauss	rad, mag	8.2 %	critique
intvol	rad, mag	8.2 %	critique
matcrs	AX=Y	4.8 %	fort
fction2	rad	3.2 %	critique

La table B.5 indique les fonctions principalement concernées par un nombre d'échange de pipeline significatif. L'unité est le pourcentage de cycles sur le nombre de cycles total durant lequel le processeur attendait. Et pour la plupart des fonctions, il s'agit de traitement d'exceptions de calculs en virgule flottante. Des bugs dans le code source en sont la source, (mauvais types de variables et calculs avec des types différents). L'écriture d'un code sans erreur est cruciale en amont du travail d'optimisation que l'on doit réaliser sur un code donnant des résultats correctes, sinon la validation des résultats et la phase de test de régression peuvent être impossibles. La génération de nombres dont la valeur est dépassée ("*overflow/underflow*")

est absolument à éviter dans une application à haute performance car elle génère de longues instructions supplémentaires de traitement des exceptions de calculs en virgule flottante.

Front-End Bandwidth MITE :

Catégorie : Front-End Bandwidth

Cette métrique représente la proportion de cycles durant lesquels le processeur attendait l'alimentation en instructions du pipeline **MITE**. Cela traduit d'éventuels problèmes dans le débit de décodage des instructions. La table B.6, présente les principales fonctions qui sont impactées.

Tableau B.6 Résumé des mesures de *Front-End Bandwidth MITE*

Fonctions	Catégories	Valeurs	Impacts
gazsig	elec	100.0 %	fort
cfsprx	maillage	69.7 %	médium
intsurf	Helmholtz	34.0 %	critique
ptgauss	rad, mag	24.7 %	critique
coeabs	rad	17.0 %	fort

Front-End Bandwidth DSB :

Catégorie : Front-End Bandwidth

Cette métrique représente la proportion de cycles durant lesquels le processeur était plutôt limité par l'alimentation en instructions du pipeline **DSB**. La majorité des instructions de notre application sont transmises par le DSB, cette métrique est donc importante.

La table B.7 fait ressortir les fonctions parmi les plus coûteuses du programme, principalement liées au calcul du rayonnement. Il y a trop d'instructions à traiter et/ou elles ne sont pas décodées à temps, ces fonctions devraient normalement être limitées par la bande passante mémoire et non pas par le débit d'instructions. Cela traduit une implémentation du modèle de calcul de rayonnement à optimiser.

Tableau B.7 Résumé des mesures de *Front-End Bandwidth DSB*

Fonctions	Catégories	Valeurs	Impacts
cfprm23	rad	100.0 %	critique
advare	maillage	95.8 %	critique
adusnb	maillage	67.6 %	fort
cfssisg	maillage	20.5 %	fort
arcsmt	modèle arc	16.2 %	critique
prfss	AX=Y	12 %	fort
surf	AX=Y	11.6 %	fort
intptp	rad	9.7 %	fort
adxlph	AX=Y	9.3 %	critique
srcrad	rad	9.3 %	critique
ibbbi	rad	8.7 %	critique
calcxy	rad, mag	8.3 %	fort
calflux	rad	7.9 %	fort
adunts	maillage	7.9 %	fort
calgrad	maillage	7.5 %	fort
sluss	AX=Y	7.2 %	fort

Front-End Bandwidth LSD :

Catégorie : Front-End Bandwidth

Le **L**oop **S**tream **D**etector, ou **LSD**, permet de prendre en compte la répétition d'instructions et de branches au cours des itérations d'une boucle. Cela permet de libérer l'unité de prédiction de branches et une partie du pipeline de décodage d'instructions en répétant une partie des instructions tant que la boucle continue. La taille des boucles (en micro-opérations) doit être adaptée à la structure du LSD. De trop nombreuses et petites boucles consécutives peuvent limiter les performances. La fraction de temps durant laquelle le CPU attendait les instructions suivantes est présentée dans la tableau B.8.

Tableau B.8 Résumé des mesures de *Front-End Bandwidth LSD*

Fonctions	Catégories	Valeurs	Impacts
cfsprx	maillage	100.0 %	faible
cfsrst	maillage	100.0 %	médium
findrg	modèle arc	100.0 %	médium
cfpdcv	fluide	100.0 %	fort
cfparc	modèle arc	100.0 %	fort
advind	maillage	95.8 %	fort
admrfn	maillage	82.1 %	fort
adyrnu	renum.	58.8 %	fort
gazsig	elec	57.5 %	fort
cfpdts	fluide	38.7 %	critique
cfsele	elec	32.9 %	fort
cfpsrc	fluide	26.1 %	fort
cfsnrl	maillage	17.9 %	fort
adxlap	maillage	17.9 %	fort
intptp	rad	17.5 %	fort
arceq2	rad	16.2 %	fort
cfsocs	maillage	9.2 %	médium

Une réécriture des algorithmes (des boucles) en mettant en commun des boucles et en réalisant des opérations différentes peut permettre d'améliorer la situation. De plus, la complexité des boucles et des algorithmes reliés à la gestion du maillage mobile et adaptatif est difficilement optimisée par le compilateur.

Branch Mispredict :

Catégorie : Bad Speculation

Afin d'accélérer le traitement des instructions en commençant des tâches à l'avance, le processeur tente de connaître à l'avance quelles seront les branches (if...else...) suivies au fil de l'exécution. L'**U**nité de **P**rédiction de **B**ranches (**BPU**) est alors mise à contribution. Cependant, lorsque des dépendances le long du fil d'exécution rendent impossibles ou trop complexes la prise de décision, l'unité BPU interroge tout d'abord son tampon d'historique et le cas échéant assume la branche comme suivie. Lorsque trop souvent de mauvaises branches sont suivies, le processeur doit ignorer une part d'instructions dans son pipeline et recharger les instructions et les données qu'il lui aurait fallu en amont. Finalement, le traitement des opérations a été ralenti car des ports (voir Figure 4.6) du scheduler sont occupés ou mis en attente inutilement par des instructions qui sont en phase d'annulation. La métrique du tableau B.9 indique la fraction de ports du CPU inutilement indisponibles à cause de mauvaises prédictions de branches.

Tableau B.9 Résumé des mesures de *Branch Mispredict*

Fonctions	Catégories	Valeurs	Impacts
gpskcd	renum.	67.8 %	fort
calcrad	rad	64.4 %	fort
aduptp	maillage	60.0 %	médium
gpskck	renum.	45.5 %	fort
srcrad	rad	45.0 %	fort
arcsmt	modèle arc	44.4 %	fort
ddot	maillage	40.1 %	fort
advord	renum.	32.9 %	faible
arcprt	modèle arc	32.8 %	fort
ibbbi	rad	32.4 %	médium
cfpgas	fluide	31.9 %	fort
intptp	rad	30.6 %	fort
sluss	AX=Y	27.9 %	fort
cfsocs	maillage	26.9 %	fort
renumber	renum.	25.5 %	fort
matcrs	AX=Y	25.5 %	fort
cfsvgc	maillage	23.3 %	médium

Pour notre cas, trop de branches sont présentes, des découpages de boucles peuvent être réalisés pour simplifier le débit d'instructions et améliorer le taux de prédictions correctes. De plus, un ré-ordonnancement des branches les plus souvent prises permet d'améliorer les performances, le test logique le plus souvent vrai pour sélectionner une des branches doit apparaître en première position dans le code source.

Machine Clears :

Catégorie : Bad Speculation

Certains événements, comme les violations d'ordonnancement mémoire (l'exécution étant réorganisée pour mettre à profit l'ILP, les données sont traitées dans un ordre différent du code source et les écritures en mémoire sont réordonnées dans le bon ordre temporel), les codes auto-modifiant, les violations d'accès en lecture dans les tableaux (accès hors des limites du tableaux,...). La métrique représente la proportion des ports du scheduler inutilement indisponibles à cause de ces types d'événements. Dans notre cas, le code n'est pas du type auto-modifiant. Cette métrique nous indique donc des problèmes potentiels liés aux accès en mémoire, les événements précités ci-dessus sont trop fréquents dans les fonctions listées. La philosophie algorithmique est à revoir pour simplifier le déroulement des accès mémoires. De plus, des algorithmes moins naïfs seraient à implémenter pour tirer partie de l'architecture processeur.

Tableau B.10 Résumé des mesures de *Machine Clears*

Fonctions	Catégories	Valeurs	Impacts
adugrp	maillage	100.0 %	fort
arcsmt	modèle arc	100.0 %	fort
asin	maillage	100.0 %	fort
cfpstr	maillage	100.0 %	fort
atan	maillage	100.0 %	fort
findrg	modèle arc	100.0 %	fort
surf	AX=Y	100.0 %	fort
siggrad	fluide	100.0 %	médium
gpskci	renum.	100.0 %	fort
intptp	rad	97.6 %	médium
advind	maillage	95.8 %	médium
adxlph	rad, mag	73.9 %	fort
temgrad	fluide	71.9 %	médium
sluss	AX=Y	71.9 %	fort
adusnb	maillage	57.6 %	médium
pow	fluide	34.0 %	fort
cfsocs	maillage	33.2 %	fort
cfpdts	fluide	30.3 %	fort

B. Mesures de performance du Back-End

Cette section se focalise plutôt sur l'accès aux données nécessaires au front-end, sur les métriques des unités de calculs logiques et en nombres flottants et sur leurs impacts lors l'exécution du code.

DTLB Overhead :

Catégorie : Memory Bound

Sous-catégorie : L1 Cache Bound

La traduction d'adresses virtuelles en adresses physiques concernant cette fois-ci les données est accélérée par les **D**ata **T**ranslation **L**ook-aside **B**uffers (**DTLB**) qui stockent en cache les traductions récemment effectuées. Des accès de données en mémoire à des adresses aléatoires

et très éloignées les unes des autres rendent inefficaces l'utilisation de ces tampons, réduisant les performances.

La table B.11 nous indique que certaines fonctions sont sujettes à des changements de page trop fréquents. La structure de données du maillage et la façon dont sont numérotées et accédées (aléatoirement) les cellules le composant sont d'un grand impact sur la vitesse d'exécution de ces fonctions. Les deux fonctions dont l'impact est critique sont les fonctions les plus coûteuses en temps de calcul du programme.

Tableau B.11 Résumé des mesures de *DTLB Overhead*

Fonctions	Catégories	Valeurs	Impacts
gazsig	elec	37.4 %	fort
cfprgl	fluide	10.3 %	critique
cfpror	fluide	9.3 %	critique
pow	fluide	5.3 %	fort

Loads Blocked by Store Forwarding :

Catégorie : Memory Bound
Sous-catégorie : L1 Cache Bound

Tableau B.12 Résumé des mesures de *Loads Blocked by Store Forwarding*

Fonctions	Catégories	Valeurs	Impacts
fluss	AX=Y	3.8 %	critique

Le processeur réorganise le déroulement du programme (Out-Of-Order execution) pour l'accélérer, cependant dans certains cas la lecture d'une donnée dépend d'une écriture amont non encore réalisée. Afin de respecter l'ordre réel des opérations, la lecture est bloquée jusqu'à ce que l'écriture soit réalisée. La métrique mesure la pénalité de performance lié à l'événement qui est à éviter fortement pour une application à haute performance. Dans notre cas, les valeurs stockées en matrice sont relues et récrites immédiatement à la suite, aléatoirement en espace, causant une dépendance dans le déroulement des instructions. Une réécriture de la fonction de factorisation peut corriger le problème, cependant une autre solution sera proposée (section 4.4).

Split Loads :

Catégorie : Memory Bound
Sous-catégorie : L1 Cache Bound

Tableau B.13 Résumé des mesures de *Split Loads*

Fonctions	Catégories	Valeurs	Impacts
fluss	AX=Y	1.3 %	critique

Les données en cache sont stockées selon des lignes de 64 bytes, lorsque des données sont à cheval entre plusieurs lignes, la lecture est faite sur plusieurs lignes à la fois au lieu d'une seule, un gaspillage de bande passante a lieu et plus d'instructions sont nécessaires pour charger les données. L'effet est très négatif et à éviter absolument pour une application HPC.

4K Aliasing :

Catégorie : Memory Bound

Sous-catégorie : L1 Cache Bound

Comme le processeur réordonne les opérations et en réalise certaines de manière concurrente. Lorsque par exemple une variable sert à deux instructions proches dans le profil d'exécution, l'une réalisant d'abord une lecture puis la deuxième une écriture :

a = b * **c**

c = e + f,

si la deuxième instruction prend moins de temps que la première, elle cherchera à écrire la valeur **c** potentiellement avant la lecture de **c** par la première instruction, ce qui est contraire à l'ordre réel du programme. Le problème est plus connu sous le nom de **WAR** (**W**rite-**A**fter-**R**ead). Pour détecter ce problème le **M**emory **O**rder **B**uffer (MOB) compare les adresses partielles de lecture et d'écriture sur seulement 12 bits ce qui crée des faux-positifs lorsque les adresses sont séparées par des multiples de 4096 bits (4K). La métrique de la table B.14 présente les pénalités de performance pour le traitement des faux-positifs.

Tableau B.14 Résumé des mesures de *4K Aliasing*

Fonctions	Catégories	Valeurs	Impacts
cfprm23	rad	100.0 %	fort
cfpmgd	maillage	100.0 %	fort
gpskcf	renum.	100.0 %	fort
cfscnd	maillage	80.5 %	fort
surf	AX=Y	75.6 %	fort
cfself	elec	67.4 %	fort
cfpidc	fluide	65.9 %	fort
cfpdcv	fluide	62.1 %	médium
fction2	rad	54.6 %	fort
calcrad	rad	42.1 %	fort
advind	maillage	36.6 %	fort
adiql1	maillage	31.0 %	médium
admrfn	maillage	28.8 %	fort
cfsnra	maillage	28.0 %	médium
srcrad	rad	28.0 %	fort
cfsrad	rad	28.0 %	fort
cfsgeo	maillage	22.8 %	fort
intvol	rad, mag	19.3 %	fort
cfsggv	maillage	15.6 %	fort
calcxy	AX=Y	14.6 %	fort
ptgauss	rad, mag	14.4 %	fort

Finalement beaucoup de fonction de MC^3 souffrent de ce problème qui a un impact très négatif sur les performances. La fonction dont la pénalité est la plus importante est aussi une des fonctions les plus coûteuse du programme. Pour ces fonctions, il faut réécrire les parties du code qui génèrent les problèmes en introduisant des variables temporaires par exemple.

FB Full :

Catégorie : Memory Bound

Sous-catégorie : L1 Cache Bound

Tableau B.15 Résumé des mesures de *FB Full*

Fonctions	Catégories	Valeurs	Impacts
adxlph	Helmholtz	100.0 %	critique
cfsggv	maillage	100.0 %	critique
cfsgeo	maillage	100.0 %	critique
cfpvcs	fluide	100.0 %	critique
cfpror	fluide	100.0 %	critique
cfpmgd	maillage	100.0 %	critique
cfpdts	fluide	100.0 %	critique
cfsocs	maillage	77.3 %	fort
calflux	rad, elec	43.5 %	fort
fluss	AX=Y	8.7 %	fort

Lorsque des données nécessaires à une instruction ne sont pas dans le cache L1, le processeur trace ce manque et fait une requête dans le **F**ill **B**uffer (FB) qui se trouve saturé si trop de données sont indisponibles. Le FB permet de répondre aux requêtes d'accès mémoire absent du cache L1, lorsque celui-ci est plein les requêtes ne peuvent être traitées, causant un ralentissement du processeur qui attend les données. Cela peut être dû à une saturation du débit du cache L1 en latence et à des mauvais patrons d'accès aux données (mauvaises structures de données et/ou mauvaises numérotations pour le maillage par exemple). La métrique indiquée à la table B.15 indique la fréquence à laquelle le FB plein a empêché les requêtes d'accès mémoire d'être satisfaites.

L2 Cache Bound :

Catégorie : Memory Bound

Tableau B.16 Résumé des mesures de *L2 Cache Bound*

Fonctions	Catégories	Valeurs	Impacts
adygog	maillage	100.0 %	critique
cfsolv	fluide	37.2 %	critique
cfpmgd	maillage	33.8 %	fort
gasrad	rad	26.1 %	critique
calgrad	rad	23.2 %	critique
ddot	AX=Y	23.0 %	critique
cfsocs	maillage	22.3 %	critique
cfsrad	rad	16.0 %	critique
arcsmt	modèle arc	13.3 %	fort
matcrs	AX=Y	12.1 %	fort
gpskch	renum.	11.5 %	fort
cfpdts	fluide	10.5 %	critique
srcrad	rad	10.0 %	fort

On s'aperçoit dans la table B.16, qu'un certain nombre de fonctions sont limitées en performance par la latence du cache de niveau 2. Lorsque les données sont absentes du cache L1, une requête est faite au cache L2 pour savoir s'il contient les données recherchées. Cette métrique représente la fraction de cycles CPU utilisés pour répondre positivement à une requête de données en provenance du cache L1. Une mauvaise gestion des données en cache L1 peut facilement saturer le cache L2. Notamment lorsque les structures de données sont trop grandes pour être contenues dans le cache L1 ou que les accès aux grands tableaux ne sont pas séquentiels. L'amélioration du taux de présence des don-

nées dans la cache L1, notamment en réutilisant les données en cache lorsqu'elles sont récentes, permet de diminuer la pression sur le cache L2 et d'augmenter significativement les performances. La performance d'un cache de niveau X est directement liée aux performances des niveaux inférieurs, il faut donc optimiser le code en ciblant d'abord les caches de plus bas niveaux. Actuellement MC^3 ne prend pas en considération l'architecture de caches.

L3 Cache Latency :

Catégorie : Memory Bound

Sous-catégorie : L3 Cache Bound

Cette métrique représente la fraction de cycles processeur durant laquelle le cache L3 répondait positivement à des requêtes de données en provenance du cache L2. L'accès aléatoire aux grands tableaux, qui ne tiennent pas dans l'espace disponible dans les caches de plus bas niveaux, force des requêtes dans les niveaux supérieurs. Le cache L2 est un miroir partiel du cache L3, si la donnée ne se trouve pas dans cette copie partielle, il faut aller la chercher soit directement dans le cache L3 avec un coût de latence important, soit faire une nouvelle copie partielle à partir de l'adresse de la donnée. Les caches chargent généralement les données 8 lignes par 8 lignes, c'est à dire 8x 64 bytes, c'est à dire des blocs de 64 nombres à virgule flottante en double précision, lors d'accès aléatoire peut être une seule valeur parmi les 64 chargées est utile au calcul, gaspillant l'espace restreint du cache (L1 32KB, L2 256KB). Au fil des itérations, la demande de données à des adresses aléatoires et éloignées les unes des autres, sature rapidement les caches en latence, ce qui gaspille aussi une part de débit très importante. Inversement, des accès par blocs contigus et séquentiels avec des algorithmes adaptés nous assurent que toutes les données chargées en cache sont réellement utiles aux calculs, ce qui peut réduire d'un facteur extrêmement important la latence et augmenter les débits de données de manière très significative, produisant une accélération très importante de la vitesse d'exécution. Par exemple,

Tableau B.17 Résumé des mesures de *L3 Cache Latency*

Fonctions	Catégories	Valeurs	Impacts
temgrad	maillage	100.0 %	médium
cfpror	fluide	100.0 %	critique
sincos	maillage	100.0 %	fort
cfscfm	maillage	100.0 %	fort
adunts	maillage	100.0 %	médium
cfsele	elec	100.0 %	fort
cfself	elec	100.0 %	médium
dengrad	maillage	100.0 %	médium
adusgp	maillage	100.0 %	médium
srcrad	rad	100.0 %	fort
cfsctn	fluide	100.0 %	médium
adugrp	maillage	100.0 %	médium
gazsig	elec	100.0 %	médium
adilct	maillage	100.0 %	fort
cfpsrc	fluide	100.0 %	fort
siggrad	maillage	100.0 %	médium
cfpbcd	fluide	100.0 %	médium
cfpvcs	maillage	100.0 %	médium
cfpmgd	maillage	100.0 %	médium

dans les solveurs linéaires, les re-numérotations des matrices, en réduisant les largeurs de bandes, ont justement pour but d'augmenter la proximité (en mémoire) ce qui accélère fortement les opérations matricielles. La table B.17 nous indique les principales fonctions réalisant de nombreux accès aléatoires (notamment à cause de l'utilisation de maillages non-structurés non re-numérotés après les adaptations successives), la plupart des fonctions de MC^3 sont limitées en performance.

Memory Bandwidth :

Catégorie : Memory Bound
Sous-catégorie : DRAM Bound

Pour cette métrique, la table B.18 présente les fractions de temps durant lesquelles la bande passante mémoire a limité les performances.

Il s'agit d'un comportement normal pour notre type d'application, toutes les données ne sont pas contenues dans les caches qui sont insuffisants pour notre volume de données correspondant à environ 800 MB. L'idéal est de saturer de manière plus efficace la bande passante mémoire en réalisant des accès contigus par blocs et en minimisant le volume total de données nécessaires aux calculs (en augmentant le ratio FLOPS/Bytes).

Tableau B.18 Résumé des mesures de *Memory Bandwidth*

Fonctions	Catégories	Valeurs	Impacts
adxifs	maillage	100.0 %	médium
siggrad	fluide	100.0 %	critique
cfpdcv	maillage	100.0 %	fort
ddot	AX=Y	100.0 %	fort
cfparc	modèle arc	100.0 %	médium
cfscrv	maillage	100.0 %	fort
adilct	maillage	100.0 %	médium
cfpsrc	fluide	78.4 %	médium
cfsadp	maillage	76.7 %	médium
cfpvcs	maillage	69.8 %	fort
cfpbcd	fluide	69.0 %	médium
cfpdts	fluide	67.3 %	médium
cfself	elec	63.0 %	médium
surf	Helmholtz	54.8 %	fort
cfpror	fluide	41.3 %	médium
intptp	rad	35.9 %	médium
gasrad	rad	33.8 %	médium
cfstmc	maillage	33.1 %	médium
cfsgeo	maillage	32.5 %	médium
ibbbi	rad	32.1 %	médium
adusnb	maillage	31.9 %	médium
cfsrad	rad	31.9 %	fort
cfpcfl	fluide	31.1 %	médium
cfsocs	maillage	29.7 %	médium

Memory Latency (LLC Miss) :

Catégorie : Memory Bound

Sous-catégorie : DRAM Bound

Lorsque des données nécessaires ne sont pas présentes dans le cache de niveau L3, une requête à la DRAM principale est faite. La transmission des données est alors régie plutôt par la latence que par le débit de la mémoire. Cette métrique représente le pourcentage de cycles durant lesquels il a fallu interroger la DRAM principale pour obtenir les données. Cela correspond plutôt à des accès inopinés et non planifiés à l'avance par les unités de prédiction du processeur.

Tableau B.19 Résumé des mesures de *Memory Latency (LLC Miss)*

Fonctions	Catégories	Valeurs	Impacts
sluss	AX=Y	100.0 %	critique
cfself	elec	100.0 %	fort
adygog	maillage	100.0 %	fort
adxifs	maillage	100.0 %	fort
intptp	rad	83.1 %	fort
cfpbcd	fluide	74.8 %	fort
cfpdts	fluide	62.8 %	fort
cfsggv	maillage	57.9 %	médium
cfstmc	maillage	30.8 %	médium
arcprt	modèle arc	12.4 %	médium
cfpror	fluide	8.7 %	critique
cfsvgc	maillage	7.6 %	médium

Au cours du déroulement des itérations, les quantités de données simultanées nécessaires aux calculs courants sont suffisamment petites pour être contenues dans le cache L3, excepté pour la fonction *sluss*. Cette fonction correspond à la résolution des systèmes matriciels avec la méthode LU qui est une méthode directe de résolution requérant une grande quantité de mémoire.

FP Vector :

Catégorie : Retiring

Sous-catégorie : FP arithmetic

La vectorisation est une forme de parallélisme. C'est un parallélisme de données où les mêmes instructions sont répétées sur des ensembles de données différents (SIMD : **S**ingle **I**nstruction **M**ultiple **D**ata). Par exemple, le calcul de la somme de deux tableaux (a = b + c) peut être accéléré en additionnant les éléments non pas uns à uns mais par blocs de X éléments consécutifs, sans perte de performance. Les unités vectorielles sont en charge des opérations vectorielles. Sur les processeurs Intel Xeon de génération Skylake, les

Tableau B.20 Résumé des mesures de *FP vector*

Fonctions	Catégories	Valeurs	Impacts
cfpmgd	maillage	100.0 %	fort
cfpvcs	maillage	71.0 %	fort
cfparc	modèle arc	50.0 %	fort
cfpdcv	fluide	10.0 %	médium
cfpdts	fluide	7.1 %	critique
fluss	AX=Y	5.8 %	critique
sluss	AX=Y	5.3 %	critique
cfsggv	maillage	4.0 %	médium
cfsgeo	maillage	1.0 %	critique
cfpror	fluide	0.3 %	critique

registres des unités vectorielles sont de 512 bits. En utilisant des nombres à virgule flottante en double précision, l'addition précédente des deux tableaux peut théoriquement être accélérée d'un facteur 8 en utilisant la vectorisation. La métrique présentée à la table B.20 nous indique le taux d'usage des unités vectorielles au cours des calculs. Dans le but de tirer partie au maximum des capacités des processeurs, il faut maximiser le taux d'usage des unités vectorielles (100 % idéalement). Lorsque la vectorisation (parallélisme intra-coeur) est combinée avec le parallélisme de threads (multi-coeurs, multi-sockets), les facteurs d'accélération sont beaucoup plus élevés. Atteindre un haut niveau de vectorisation est extrêmement crucial pour notre application de calcul à haute performance, des solutions spécifiques seront détaillées dans la section 4.4.

Pour les fonctions non citées dans la table B.20, les unités de calcul en virgule flottante ne sont pas utilisées du tout, les unités scalaires sont alors les seules utilisées pour les calculs arithmétiques. Un fort potentiel de puissance n'est pas utilisé, en désactivant lors de la compilation les instructions vectorielles, la différence de vitesse est de seulement $4 - 5\%$. La différence de performance ne provient donc pas d'une utilisation inefficace des unités vectorielles mais d'un manque d'instructions vectorielles. Les fonctions de MC^3 les plus coûteuses ne sont pas du tout représentées dans la table B.20. La ré-écriture d'un code plus explicite pour l'interprétation par le compilateur pourrait favoriser l'auto-vectorisation du code, sinon l'ajout (dans le code source) de conseils (pragmas) et d'informations supplémentaires pour le compilateur renforceront ses capacités à saisir les opportunités d'optimisation automatique.

Divider :

Catégorie : Core Bound

Toutes les opérations arithmétiques ne consomme pas la même quantité de ressources et de temps d'exécution. Les divisions et les racines, toutes les deux réalisées par les unités DIV, sont beaucoup plus coûteuses que les opérations logique, les additions/soustractions et les multiplications. Cette métrique, présentée à la table B.21, représente la fraction des cycles durant lesquels l'unité de division DIV était active.

Tableau B.21 Résumé des mesures de *Divider*

Fonctions	Catégories	Valeurs	Impacts
arcsmt	modèle arc	100.0 %	fort
siggrad	fluide	100.0 %	médium
surf	Helmholtz	100.0 %	fort
intptp	rad	93.2 %	fort
cfsprx	maillage	69.7 %	médium
gasrad	rad	67.6 %	médium
cfpmgd	maillage	63.0 %	médium
calcxy	AX=Y	62.7 %	médium
calflux	rad	61.4 %	médium
cfpdcv	fluide	59.0 %	médium
ptgauss	rad, mag	53.6 %	critique
icompo	fluide	46.5 %	médium
cfpror	fluide	38.6 %	critique
cfptsi	fluide	33.6 %	critique
intvol	rad, mag	31.7 %	critique
fmod	math	31.1 %	fort
intsurf	Helmholtz	29.3 %	fort
cfpgas	fluide	28.5 %	critique
cfsgeo	maillage	27.5 %	fort
cfstri	maillage	26.6 %	fort
cfprgl	fluide	24.9 %	critique
cfscfm	maillage	24.1 %	médium
cfpbcd	fluide	23.0 %	médium
adiql1	maillage	20.4 %	médium
fluss	AX=Y	14.6 %	fort
cfsnrl	maillage	14.2 %	médium
matcrs	AX=Y	10.9 %	fort
log	fluide	10.7 %	fort
cfsvgc	maillage	9.3 %	médium
adxlap	maillage	8.7 %	fort
pow	fluide	8.6 %	fort
cfsggv	maillage	4.5 %	faible